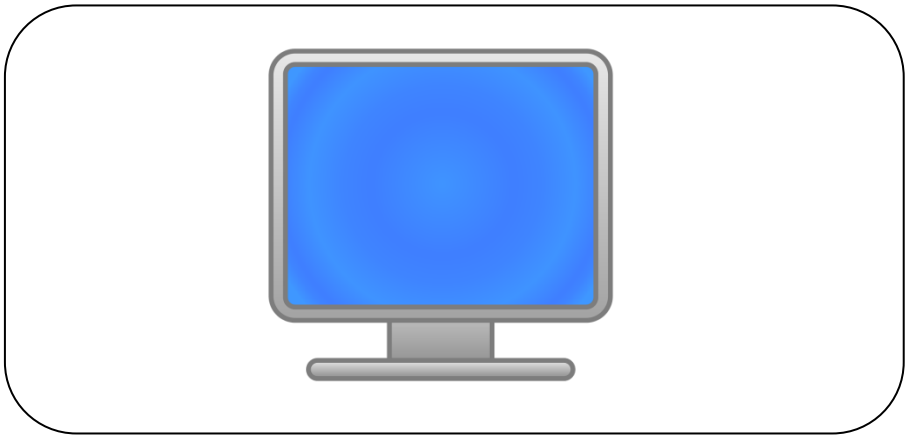
Métiers de l'informatique, vocabulaires et thèmes pour les métiers du numérique orientés (ingénieur en informatique)

Tome 1 : les notions

Par Dimitri PIANETA



Edition 2020 version 1

Introduction

Ce tome 1 de « **Métiers de l'informatique, vocabulaires et thèmes pour les métiers du numérique orientés (ingénieur en informatique** est orienté sur les connaissances des métiers du numériques et les bases pour comprendre les éléments théoriques que nous pouvons attendre un peu partout et dans les entretiens de recrutement.

Ce tome 1 est plus orienté bureau d'étude ingénieur en informatique. Je n'ai pas pu tout mettre dans ce premier tome.

J'ai mis l'ensemble des connaissances en synthèse qui me semblait intéressant à connaître.

Je vous souhaite une bonne lecture.

Dimitri PIANETA

Attention, ce document n'est pas une bible sur l'ensemble du numérique. Je suis mis dans ma situation de développeur informatique et scientifique du numérique.

Edition Juillet 2020, France

SOMMAIRE

Partie 1 : les métiers du numérique

Les acteurs de la fonction Informatique

Les métiers de l'informatique s'organisent autour de deux axes principaux : l'un constitué par le monde des prestataires (fabricants et intermédiaires) et d'autre, par celui des utilisateurs.

Les principaux acteurs du monde de l'informatique sont résumés sur ce tableau :

Fabricants	Intermédiaires	Utilisateurs
Constructeurs de matériels informatiques et loueurs	Sociétés de services et conseil en SI[1]	Grands comptes (publics et privés)
Editeurs de logiciels	Distributeurs : - Revendeurs - Grossistes - VPCistes[2] - Commerce électronique - Places de marché	PME[3] ETI[4] TPE[5] Particuliers

De fait, le monde des intégrateurs-conseil est aujourd'hui composé de quatre types d'acteurs :

- Les **grands intégrateurs** qui occupent une position d'ensembliers, en intégrant en particulier une activité Conseil en système d'information.
- Des **SSII[6] de taille plus réduite** qui proposent des prestations plus « verticalisées » (par secteurs d'activité, par type de prestations et/ou par compétences techniques) et dont l'activité conseil est en général moins développée.
- **Quelques cabinets de conseil** de petite ou moyenne taille qui gardent leur indépendance par rapport aux sociétés de service.
- Un certain nombre de **consultants indépendants** qui exercent leur activité auprès de niches (PME-PMI, professions libérales, prestataires de formation par exemple) ou sur des prestations souvent très pointues (sécurité, langages, matériels…).

[1] Système information
[2] Entreprises de vente par Correspondance. Le terme général est vépéciste.
[3] Petites et les moyennes entreprises (**PME**) de 10 à 249 salariés.
[4] Entreprises de taille intermédiaire (**EDI**) comptent de comptent entre 250 et 4 999 salariés.
[5] Très petites entreprises **(TPE)** de moins de 10 salariés.
[6] **Société de services et d'ingénierie en informatique** ce terme a été modifié actuellement par ESN qui signifie **entreprise de services du numérique.**

Les quatre domaines d'application de l'informatique sont résumés par ce tableau :

Domaines informatiques	Applications
L'informatique de gestion	Elle s'applique à l'organisation des informations dans l'entreprise : l'administration de l'entreprise, la gestion commerciale, les ressources humaines, mais aussi la facturation, la gestion de stocks et des commandes… Dans ce domaine, les informaticiens ont de plus en plus fréquemment recours aux progiciels[7] disponibles (PGI ou ERP), utilisés après paramétrage pour les adapter aux spécificités de l'entreprise.
L'informatique scientifique	Elle s'applique au calcul dans le domaine des sciences exactes, à la modélisation, aux essais, à la recherche fondamentale, à l'informatique en temps réel… Les experts dans ce domaine voient le champ de leurs travaux s'étendre à la dimension du monde visuel. Ces spécialistes sont généralement de formation ingénieur et s'appuient sur une forte culture mathématique.
L'informatique industrielle	Elle couvre le champ des applications logicielles, destinées au pilotage des chaînes de production et aux produits industriels. Elle concerne également la simulation et les interfaces hommes-machines(IHM). Les logiciels d'informatique industrielle sont utilisés par des ingénieurs ou techniciens.
L'informatique technique	Elle est appliquée à la programmation des logiciels intégrés à des produits : dans le domaine des télécommunications (téléphone portable, télévision numérique), des transports, de l'automobile, de l'aéronautique… Les nouvelles technologies (Internet, communication sans fil et électronique embarquée) se démocratisent et se déploient sous des formes plus ergonomiques en rassemblant sons, images et textes sur un seul support.

[7] Ensemble de programmes prêts à l'emploi et mis sur le marché par un éditeur. On définit PGI = ERP = progiciel de gestion intégrée en anglais Enterprise Resource Planning. Les ERP sont un ensemble de modules logiciels intégrés autour d'une base de données unifiée pour des applications comme les ventes et la distribution, la comptabilité financière, la gestion des investissements, la planification de la production, la maintenance des installations et les ressources humaines.

Les deux modalités d'intervention de la ESN chez l'utilisateur : la régie et le forfait

1/ La Régie

Le principe de la régie est de mettre des informaticiens à la disposition des entreprises utilisatrices. Le client garde la maîtrise d'œuvre du projet et intègre l'expert détaché par le ESN au sein de ses propres équipes. Le recours à des sociétés de service est un élément de flexibilité et d'adaptabilité essentiel : quel que soit le projet informatique de l'entreprise, elle trouvera à sa disposition du personnel compétent qu'elle n'aura pas à former.

Quand aux ESN, la régie constitue pour elles une source de revenus régulière.

Pour la cadre informatique, cette formule présente l'avantage d'un travail au cœur de l'entreprise utilisatrice. C'est aussi un bon tremplin pour le jeune diplômé à la recherche de son premier emploi. En revanche, les évolutions de carrière, de rémunération et de responsabilités sont parfois difficiles à obtenir.

2/ Le forfait

Le principe du forfait est de sous-traiter tout ou partie d'un projet à un prestataire de service. L'entreprise cliente fixe les contraintes en matière d'objectifs, de délais et de coûts du projet.

Comme dans le cas de la régie, l'entreprise, grâce à ses prestataires de service, dispose de personnels formés sur délégation d'un projet à un prestataire de service accélère sa mise en place. Le prix de la prestation est plus élevé que celui de la régie.

Ce mode de contrat permet à l'informaticien de travailler dans les locaux de son employeur, de collaborer au sein d'une équipe projet homogène (tous salariés de ESN) et d'intervenir sur des dimensions assez qualifiées du système d'information.

Cartographie des métiers par famille

Direction/stratégie

N°1 – Directeur des systèmes d'information
N°2 – Directeur technique
N°3 – Directeur des études
N°4 – Directeur de département Conseil et SI

Développement et intégration

N°5 – Ingénieur d'étude et développement
N°6 : Ingénieur de production
N°7 : Ingénieur développement logiciel
N°8 : Ingénieur en informatique industrielle
N°9 : Chef de projet fonctionnel (web)
N°10 : Chef de projet maîtrise d'oeuvre
N°11 : Chef de projet technique (web)
N°12 : Architecte infrastructures
N°13 : Architecte WEB

Production/exploitation/maintenance

N°14 : Responsable d'exploitation
N°15 : Responsable de la maîtrise d'ouvrage bancaire/MOA
N°16 : Responsable de parc informatique
N°17 : Responsable Du SIRH
N°18 : Responsable informatique
N°19 : Analyste d'exploitation

Système/réseau/données

N°20 : Administrateur de bases de données
N°21 : Administrateur réseau
N°22 : Ingénieur système
N°23 : Ingénieur sécurité WEB
N°24 : Responsable sécurité informatique
N°25 : Webmaster

Conseil en SI/maîtrise d'ouvrage

N°26 : Consultant fonctionnel
N°27 : Consultant informatique décisionnelle/ BIG DATA
N°28 : Consultant intégrateur de progiciel
N°29 : Consultant technique
N°30 : Responsable système d'information métier
N°31: Urbaniste architecte fonctionnel système d'information
N°32 : DATA SCIENTIST – DATA MINER

Commercial/marketing

N°33: Directeur commercial
N°34: Ingénieur d'affaires
N°35 : Ingénieur commercial
N°36: Chef de produit technique
N°37: Chef produit web/mobile
N°38: Ingénieur avant-vente

Direction/stratégie

N°1 : Directeur des systèmes d'information (DSI)

Autres intitulés :

- DIRECTEUR DE L'ORGANISATION ET DES SYSTEMES D'INFORMATION (DOSI)
- DIRECTEUR INFORMATIQUE
- télécommunications (DI)

On peut traduire en anglais par : *Chief information officer* et *Business information officer*

Définition : **Le directeur des systèmes d'information a pour mission de définir et mettre en œuvre la politique informatique en accord avec la stratégie générale de l'entreprise et ses objectifs de performance. Il doit garantir la continuité du service informatique fourni aux utilisateurs et anticiper les changements et leurs impacts métiers sur le système d'information.**

Activités principales :

Elaboration de la stratégie et de la politique informatique

- Définir avec la direction générale et les membres du comité de direction les orientations stratégiques en matière d'informatique et de télécommunications.
- Définir la politique en matière de sécurité informatique : identification avec la direction générale des informations sensibles et des risques, proposition des mesures à prendre.
- Recueillir et étudier les besoins exprimés par les directions métiers de l'entreprise.
- Evaluer et préconiser les investissements informatiques correspondant aux besoins métiers exprimés en tenant compte de leur efficacité et de la maîtrise des risques.
- Assurer une veille technologique et juridique sur les évolutions du secteur de l'entreprise en matière de systèmes d'information.
- Anticiper les changements et orienter les choix de la direction générale en matière de technologies de l'information (schéma directeur informatique).
- Concevoir une organisation optimale des flux d'information de l'entreprise.
- Assurer l'adéquation entre les besoins des clients de l'entreprise, la stratégie de la société et les outils informatiques.

Profil :

Diplômes requis

- Formation de niveau Bac + 5 : master en informatique
- Ecole d'ingénieurs (informatique, télécoms, généraliste)
- Eventuellement diplôme d'école de commerce complété par un master en gestion des systèmes d'information

Durée d'expérience

Une expérience de cinq à dix ans en tant que responsable de service ou d'une direction informatique (informatique, télécoms ou métier) dans une SSII ou dans une entreprise utilisatrice est nécessaire.

Compétences requises :

Compétences techniques

- Connaissance large des systèmes d'information et de leurs évolutions
- Connaissance des applications et des technologies utilisées dans l'entreprise, des principaux langages informatiques et systèmes d'exploitation
- Maîtrise des normes de sécurité et de l'actualité des risques mondiaux en matière de sécurité
- Bonne connaissance du marché de la sous-traitance : éditeurs, SSII, cabinets de conseil… et gestion de la relation avec la sous-traitance
- Très bonne connaissance des métiers et de l'organisation de l'entreprise, des besoins de toutes les autres directions
- Bonne culture économique, en particulier sur le secteur d'activité de l'entreprise
- Certaines bases en finance et en contrôle de gestion, pour piloter la gestion du budget informatique et favoriser le dialogue avec la direction financière
- Bonne maîtrise des méthodologies de gestion multi projets
- Maîtrise de l'anglais technique

Aptitudes professionnelles

- Sens de l'anticipation pour mettre en œuvre des solutions innovantes, en fonction de la stratégie de l'entreprise
- Organisation, rigueur et autonomie : la fonction de dirigeant peut parfois s'accompagner d'une certaine solitude. Dans ce contexte, il est important que le DSI soit autonome et puisse avancer avec sérénité
- Adaptabilité car le DSI doit garder à l'esprit que le secteur informatique évolue toujours et qu'une veille technologique est primordiale pour maintenir le SI en état de fonctionnement et l'optimiser
- Qualités relationnelles et sens de l'écoute dans ses rapports fonctionnels et hiérarchiques
- Sens de la négociation avec les collaborateurs internes (définir les objectifs, participés au processus de recrutement) et les prestataires (obtenir le produit ou le service offrant le meilleur rapport qualité/prix pour l'entreprise)
- Bonnes compétences rédactionnelles pour la formalisation de clauses contractuelles et la présentation à la direction générale des évolutions nécessaires
- Pédagogie car le DSI doit expliquer certaines de ses décisions et démontrer leur bien-fondé devant des non-initiés (DG, DAF…) ; il doit, au préalable expliciter les évolutions et les progrès attendus.
- Le poste de DSI nécessite une implication très importante dans son travail car l'activité de management est chronophage et le titulaire doit se réserver des moments de réflexion pour adapter la politique informatique aux besoins des utilisateurs.

N°2 : DIRECTEUR TECHNIQUE

Autre intitulé :

Directeur de la production

Définition :

Le directeur technique informatique a pour mission de gérer le cycle de développement et de déploiement du produit et/ou du service en s'appuyant sur des ressources internes et externes.

Activités principales :

Définition de la stratégie d'innovation de l'entreprise

- Étudier le positionnement technique de l'entreprise (technologies engagées, qualification de l'environnement technologique, etc.).
- Travailler en interface avec la direction marketing pour appréhender les évolutions des besoins des clients et les transformer en orientations techniques.
- Assurer et/ou piloter une veille technologique (langages de développement, infrastructures techniques dans ces environnements...).
- Nouer et entretenir des relations avec des partenaires technologiques de l'entreprise.
- Définir la stratégie d'acquisition de brevets de l'entreprise.

Participation aux phases d'avant-vente

- Participer aux phases de réponse aux appels d'offre.
- Intervenir ponctuellement auprès du client lors de la phase de présentation de l'offre sur le plan technique.
- Superviser la phase de rédaction des spécifications techniques et les valider.

Management des phases de déploiement de l'offre

- Organiser, planifier et contrôler les phases de développement.
- Piloter le choix et l'intervention d'éventuels prestataires extérieurs.
- Encadrer les équipes de développement interne ; en particulier stimuler les processus de formation et d'autoformation de ses équipes et gérer les recrutements.
- Contribuer à la capitalisation des bonnes pratiques et à l'amélioration de la qualité dans les phases de développement.
- Superviser les phases de recette, de documentation technique et de maintenance des applications.
- Piloter les opérations de consultance et de support client (ces équipes étant le plus souvent sous la responsabilité du directeur technique).

Profil :

Diplômes requis

- Écoles d'ingénieurs (informatique, télécoms, généraliste)
- DESS / DEA informatique
- Diplôme de type Bac +4 en informatique : MIAGE, IUP informatique, maîtrise informatique, ingénieur maître...

Durée d'expérience

Ce poste requiert au minimum six ans d'expérience.

Compétences requises :

Compétences techniques

- Bonne culture générale informatique et connaissance de l'ensemble du système d'information de l'entreprise
- Très bonne connaissance de l'environnement technique propre à l'entreprise (par exemple *Business Intelligence*, CRM, Internet, Sécurité…)
- Connaissance des offres logicielles associées
- Très bonne connaissance du développement logiciel : langages de développement et méthodes de développement et de projets associés
- Connaissance des systèmes d'exploitation (Windows, Unix, Linux, etc.)
- Maîtrise des processus d'innovation et de gestion du changement
- Compétences en gestion (méthodes de choix d'investissement, élaboration et suivi d'un budget, contrôle de gestion, etc.)
- Bonne connaissance des méthodologies projet (UML, cycles de développement de projet…)
- Maîtrise de l'anglais technique

Aptitudes professionnelles

- Bonne vision stratégique et capacité à fixer des objectifs prioritaires aux équipes et aux prestataires
- Capacité à prendre, parfois très rapidement, des décisions qui engagent l'entreprise
- Qualités d'organisation, de planification et de rigueur pour piloter les projets
- Autorité managériale pour prendre en charge une équipe parfois assez importante et éclatée (de nombreux services, des équipes internes et externes, en régie ou au forfait)
- Curiosité, goût du changement et bonne capacité d'adaptation à un environnement évolutif et complexe
- Force de conviction afin de convaincre en interne de la pertinence des solutions techniques choisies
- Capacité à stimuler mais aussi à canaliser la créativité des équipes
- Calme et sang-froid afin de résister à la pression des utilisateurs internes et externes, diplomatie et pédagogie afin d'apaiser leurs inquiétudes et leurs doutes

N°3 : DIRECTEUR DES ETUDES

Autres intitulés :

- Responsable des études informatiques
- Directeur études et développement informatiques
- Responsable organisation et études S.I.

Définition :

Le directeur des études recueille et analyse les besoins des directions métiers de l'entreprise en matière de système d'information, et pilote le développement, l'intégration et la maintenance des solutions applicatives.

Activités principales :

Définition de la stratégie informatique de l'entreprise

- Recueillir les besoins des utilisateurs et les traduire en besoins informatiques.
- Evaluer le budget nécessaire et suivre sa réalisation.
- Participer aux choix des orientations stratégiques et à la rédaction de schémas directeurs.
- Définir les orientations technologiques et méthodologiques.
- Mettre en place les principes de consultation des prestataires extérieurs.
- Définir la répartition entre internalisation et externalisation.
- Définir la gouvernance des études informatiques et y faire adhérer les différents acteurs.

Management de projets

- Superviser et coordonner le travail des directeurs de programmes et/ou de projet et animer les équipes internes et/ou externes.
- Définir les objectifs du service en termes de qualité, performances, coûts, délais et sécurité.
- Participer à la contractualisation des objectifs avec les donneurs d'ordre (métiers).
- Suivre les appels d'offres, superviser leur dépouillement et valider le choix des prestataires.
- Superviser les plannings, les charges et les budgets des projets.
- Assurer l'interface et la communication entre les métiers de l'entreprise, les études informatiques et la production.
- Assurer le reporting de l'activité auprès du directeur informatique et les comités de pilotage.

Management d'équipes

- Superviser ou assurer le recrutement de nouveaux collaborateurs, en tant que salariés de l'entreprise ou en tant que prestataires (SSII).
- Gérer les moyens humains nécessaires à la mise en place et au bon déroulement du projet et arbitrer les demandes des directeurs de projets.
- Evaluer la performance de ses équipes.
- Contrôler l'efficacité des réalisations et des prestations des sous-traitants tout au long de la relation contractuelle.

Veille technologique et marché

Assurer une veille technologique afin d'être force de proposition et maintenir l'adéquation des ressources informatiques aux besoins de l'entreprise et de guider les choix stratégiques vers les meilleures solutions.

Profil :

Diplômes requis

- Formation de niveau bac +4/5 (master) spécialisée en informatique, réseaux et télécommunications, éventuellement complétée par une formation de type IAE
- École d'ingénieurs (informatique, télécoms, généralistes..), éventuellement complétée par une formation de type IAE

Durée d'expérience

Ce poste nécessite au moins dix ans d'expérience notamment dans la direction de projet.

Compétences requises :

Compétences techniques

- Excellente compréhension de l'environnement et des activités de l'entreprise, des besoins et des contraintes des utilisateurs
- Bonne culture informatique incluant une bonne connaissance de l'architecture S.I., des méthodes et principes de développement d'une application, des infrastructures informatiques et de types d'applicatifs ou d'ERP (progiciels intégrés)
- Connaissance des principaux prestataires du marché informatique (éditeurs, sociétés de services...)
- Bonne connaissance du domaine d'application métier concerné par les projets pris en charge (afin d'asseoir sa légitimité auprès de la maîtrise d'ouvrage et des utilisateurs)
- Connaissance des modalités d'élaboration d'un schéma directeur
- Connaissance des outils de gestion de projet et de PMO (MS Project)
- La maîtrise de l'anglais peut être nécessaire pour des projets internationaux ou au sein d'entreprises étrangères
- Bonnes compétences en gestion (calcul de ratios et de rentabilité, gestion budgétaire, trésorerie, retour sur investissement…)
- Notions juridiques dans les domaines des contrats et du droit informatique

Aptitudes professionnelles

- Esprit de synthèse de manière à avoir une vision globale des projets et pour identifier les priorités de développements informatiques de l'entreprise
- Capacité à vulgariser des sujets techniques complexes afin de permettre aux membres des comités de pilotage / ou de direction de s'approprier les sujets et de réaliser des arbitrages de manière éclairée
- Très bonnes qualités relationnelles et de communication afin d'assurer une collaboration efficace avec la maîtrise d'ouvrage et les directions métiers
- Diplomatie pour concilier des intérêts parfois divergents : ceux des métiers et ceux techniques et financiers

- Sens commercial pour « vendre » en interne ou en externe des solutions, notamment dans le cadre des contrats de service
- Sens de la négociation, notamment dans le cadre des relations avec les fournisseurs
- Force de persuasion car le directeur des études doit parfois convaincre ses interlocuteurs du bien fondé de ses choix
- Rigueur et méthode afin de pouvoir suivre l'articulation des différents projets et leur avancement
- Capacité à intégrer les contraintes et enjeux de nouvelles technologies en s'appuyant sur son équipe ainsi que sur des experts
- Qualités d'animation d'équipes : écoute, dialogue pour à la fois animer et valoriser le travail de son équipe
- Capacité à motiver, à stimuler les équipes et à leur faire comprendre les impératifs en termes de délais sans pour autant créer des tensions défavorables à l'atteinte des objectifs assignés
- Capacité à prendre des décisions difficiles dans le cadre de difficultés dans le déroulement des projets
- Très bonne résistance au stress car les projets informatiques sont toujours soumis à des contraintes de délais et sont souvent considérés comme stratégiques pour l'entreprise et les directions métiers

N°4 : DIRECTEUR DU DEPARTEMENT CONSEIL et SI

Autres intitulés

- Directeur de l'activité conseil
- Directeur de *business unit*
- Directeur du département services
- Directeur associé

Définition :

Le directeur du département Conseil et SI a pour mission de définir une offre de service et de la déployer sur les plans marketing, commercial et technique à l'aide d'équipes placées sous sa responsabilité.

Activités principales :

Conception et positionnement de l'offre

- Analyser, structurer et décrypter l'environnement concurrentiel.
- Définir l'offre de service, mettre en exergue ses avantages concurrentiels, la qualifier fonctionnellement et techniquement.
- Définir la cible commerciale (taille d'entreprise, secteurs d'activité, implantation géographique) et les canaux de commercialisation directs et indirects (prescripteurs).
- Déterminer les moyens humains et les modalités de livraison de la prestation (répartition entre régie et forfait).

Mise en œuvre de la stratégie de développement de l'activité conseil

- Initier, coordonner et s'investir personnellement dans les actions de communication, de relations presse et de relations publiques autour de l'activité conseil et de la prestation commercialisée : organisation de petits déjeuners et de séminaires, participation aux salons et événements.
- Coordonner les actions de développement en interface avec les équipes commerciales.
- Assurer des relations à haut niveau avec les prescripteurs de l'offre commercialisée (éditeurs, sociétés de service, sociétés de conseil, etc.).
- S'investir personnellement dans certaines phases de l'avant-vente.
- Valider les réponses aux appels d'offre.

Animation et coordination des équipes projet

- Définir le plan de charge des équipes, superviser les plannings et gérer les inter contrats (pour les collaborateurs n'étant pas en mission).
- Recruter, encadrer et motiver les équipes en charge du "*delivery*" (c'est-à-dire réalisant la prestation d'intégration : ingénieur d'étude, chef de projet, directeur de projet) et en charge du conseil (consultant junior et senior).
- Assurer la relation auprès du client à très haut niveau.
- Superviser et coordonner l'ensemble des projets et vérifier l'adéquation de la prestation au cahier des charges défini initialement.

Profil :

Diplômes requis

- Écoles d'ingénieurs (informatique, télécoms, généraliste)
- Écoles supérieures de commerce ou 3e cycle en gestion (mastère)
- DESS / DEA informatique
- Écoles supérieures de commerce, ou formations en gestion des NTIC (INT Management, DESS NTIC...)

N.B. : Les doubles formations école d'ingénieurs/école de commerce sont particulièrement appréciées. Les grandes écoles de commerce et d'ingénieurs sont souvent privilégiées.

Durée d'expérience

Le directeur du département Conseil dispose d'une expérience minimum de six ans.

Compétences requises :

Compétences techniques

- Forte légitimité et expertise dans le domaine de la prestation (il s'agira pour le directeur de département d'être considéré comme un expert soit du domaine fonctionnel relatif à la prestation (achats, ressources humaines, finances, production, etc.), soit du domaine technique (infrastructure, sécurité, réseaux, portails, etc.), soit du domaine technico-fonctionnel (par exemple ERP ou *business intelligence*)
- Connaissance panoramique des solutions logicielles proposées sur le marché, qu'elles soient concurrentes ou complémentaires
- Excellent carnet d'adresses, aussi bien auprès des prescripteurs (sociétés de service, éditeurs de logiciels, sociétés de conseil), des leaders d'opinion (presse spécialisée, portails Internet spécialisés, associations professionnelles) et des utilisateurs
- Bonne culture économique et financière
- Maîtrise de l'anglais, car une partie des acteurs (prescripteurs ou décideurs) est basée ailleurs qu'en France

Aptitudes professionnelles

- Qualités conceptuelles et analytiques pour concevoir et structurer une offre
- Force de persuasion et sens de la négociation pour convaincre en interne (notamment la direction générale, les équipes marketing et commercial) et en externe (les clients et les prescripteurs)
- Diplomatie pour gérer la dimension «politique» des projets
- Rigueur, organisation et puissance de travail afin de pouvoir mener en parallèle plusieurs projets
- Bonne résistance au stress : forte pression des clients, importante charge de travail

Développement et Intégration

N°5 : Ingénieur cloud et virtualisation

Autres intitulés :

- Ingénieur solutions cloud
- Ingénieur infrastructures cloud
- Chef de projet cloud

Définition :

L'ingénieur cloud et virtualisation est en charge du déploiement, du stockage et de la gestion des données sur des serveurs situés dans des data centers hors de l'entreprise. Il intervient dans le cadre de la mise en service de plateformes informatiques et de traitements distants, à la demande et mutualisés.

Activités principales :

Analyse des besoins et veille technologique

- Recueillir l'information nécessaire et étudier les besoins en infrastructures.
- Préconiser les solutions et des architectures cibles en réponse aux besoins des utilisateurs et les conseiller sur les choix finaux.
- Rédiger le cahier des charges contenant les spécifications techniques des équipements.
- Assurer une veille technologique pour garantir l'optimisation des ressources en termes d'infrastructures.

Intégration dans une architecture cloud

- Mettre en place un pilote.
- Assurer la migration vers des solutions virtualisées dans le *cloud.*
- Configurer et dimensionner les plateformes en fonction des performances requises.
- Industrialiser les architectures cibles et configurer les équipements.
- Définition des accès aux ressources via des API.
- Mettre en place le système de gestion des droits des différents utilisateurs.
- Mettre en place des procédures et outils de surveillance permettant de garantir la haute disponibilité des infrastructures.
- Veiller à la sécurité des accès et à la fiabilité des solutions déployées.
- Participer aux phases de validation technique lors des mises en production ou des évolutions.
- Gérer les relations avec les éditeurs de virtualisation (gestion des licences et du contrat de maintenance).

Administration et maintenance du système virtualisé

- Maintenir en condition opérationnelle les infrastructures virtualisées en diagnostiquant les pannes et les dysfonctionnements liés aux problèmes d'infrastructures.
- Réparer les pannes et les dysfonctionnements.
- Ajouter, supprimer, sauvegarder les machines virtuelles.

- Assurer une maintenance évolutive et corrective en fonction des grandes évolutions technologiques (notamment les changements de versions).
- Optimiser les performances des systèmes d'exploitation (tuning).
- Installer les patches de sécurité et administrer le firewall.
- Assurer un support technique de second niveau.

Profil :

Diplômes requis

- Formation de niveau Bac +5 (Master 2) spécialisée en informatique, réseaux et télécommunications (par exemple masters et masters spécialisés de l'Isep, de l'Insset, de l'Istescia)
- Écoles d'ingénieurs (informatique, télécoms, généralistes...), notamment Télécom Lille 1, Telecom Paris Tech, Telecom Bretagne, Telecom Paris Sud, Eurecom, Epita, Isep, Itescia...

Durée d'expérience

Ce poste s'adresse de préférence aux cadres ayant au moins deux à trois ans d'expérience comme ingénieur système. Néanmoins de jeunes diplômés avec un stage dans le domaine de la virtualisation des systèmes peuvent aussi être recrutés.

Compétences requises :

Compétences techniques

- Maîtrise voire expertise des logiciels de l'infrastructure technique notamment des systèmes d'exploitation Linux et Windows et des outils de virtualisation (de type VMWare)
- Bonne connaissance des langages systèmes de type Python
- Bonnes connaissance de l'administration réseaux et télécoms
- Bonne maîtrise des technologies Internet : protocoles de sécurité, protocoles Internet…
- Bonne connaissance de l'architecture et des fonctionnalités du SI de l'entreprise
- Connaissance des principales solutions cloud du marché
- Connaissance des bases de données et des outils de stockage
- Connaissance des normes et procédures de sécurité et des outils associés
- Bonne culture informatique : principaux langages et outils de développement
- Bonne connaissance des clients internes de la DSI : les principaux métiers de l'entreprise
- Maîtrise de l'anglais technique

Aptitudes professionnelles

- Rigueur, organisation et méthode : afin de mettre en place les infrastructures en respectant des méthodes et des règles de qualité définies
- Capacité d'analyse pour trouver la solution la plus rapide et la plus adaptée
- Bonnes qualités relationnelles dans les rapports avec les utilisateurs internes et externes
- Capacité d'anticipation et de prises d'initiative pour assurer une veille technologique efficace
- Adaptabilité et curiosité technique car les évolutions technologiques sont rapides et doivent être assimilées afin de pouvoir optimiser l'existant
- Polyvalence, car il doit comprendre les façons de travailler différents utilisateurs des infrastructures cloud

N°6 : Ingénieur de production

Autres intitulés :

- Ingénieur d'exploitation
- Analyste d'exploitation

Définition :

L'ingénieur de production informatique garantit le fonctionnement technique des moyens de production. Il préconise et met en œuvre les solutions méthodologiques et techniques permettant d'optimiser la production informatique.

Activités principales :

Définition des procédures d'exploitation

- Participer à l'étude et à l'analyse des besoins de l'entreprise en matière de système d'exploitation.
- Réaliser l'analyse des contraintes d'exploitabilité.
- Assurer et optimiser l'industrialisation de la production.
- Définir les procédures et diffuser les consignes de mise en production.
- Définir et mettre en place les outils de normalisation, d'automatisation et de sécurisation de la production.
- Mettre en place et diffuser les indicateurs (performance, coût, délai), les normes (ITIL) et les procédures d'exploitation.
- Maintenir en condition opérationnelle (MCO) les moyens en veillant au respect des normes et des critères de performance.
- Participer à la définition du plan de reprise des activités (PRA).
- Assurer une veille technologique en matière de méthodes et outils de production.

Réalisation de l'intégration technique d'applications

- Définir les plans de mise en production (PMEP).
- Installer les logiciels et les matériels retenus et configurer les postes de travail.
- Définir les tâches de servitude : alarmes, ordonnancement, sauvegardes.
- Planifier les traitements (batch).
- Déployer et faire migrer les serveurs avec l'ingénieur système.
- Intégrer les nouvelles applications au système d'exploitation existant.
- Assurer la mise en activité réelle d'une correction ou d'une mise à jour d'une composante logicielle ou matérielle après sa validation en pré-production (RUN).
- Mettre en place un programme de tests des nouvelles installations.
- Assurer une surveillance en temps réel des systèmes et des applications.
- Analyser et gérer les incidents d'exploitation, opérer la restitution ou le retour en arrière et assurer les opérations de maintenance.
- Piloter les relations avec les prestataires.

Support aux utilisateurs

- Assurer un support technique aux utilisateurs (2ème et 3ème niveau).
- Intervenir sous contraintes de délais lors des incidents d'exploitation (analyse des incidents, diagnostic et résolution des incidents).
- Anticiper les besoins des utilisateurs et les évolutions du système.

Analyse des rapports techniques

- Collaborer à la définition ou à l'amélioration des procédures sécurité.
- Etablir la documentation technique.
- Veiller à l'efficacité permanente des systèmes de sécurité.
- Suivre les incidents d'exploitation, en analyser les causes et prendre les mesures correctives associées.

Profil :

Diplômes requis

- Diplôme de niveau Bac +2en informatique : BTS/DUT informatique
- Diplôme de niveau Bac +4/5 en informatique : IUP informatique, master en informatique
- Ecole d'ingénieurs (informatique, télécoms, généralistes)

Durée d'expérience

Ce poste peut être proposé à des débutants mais s'adresse généralement à des cadres ayant deux à trois ans d'expérience professionnelle

Compétences requises :

Compétences techniques

- Connaissances des matériels, des logiciels, des normes de fichiers, des systèmes d'exploitation (scripts Shell) des applications, et des liaisons inter applications
- Connaissance des procédures de transmission de données
- Connaissance des systèmes d'exploitation et bases de données associées
- Connaissance de l'architecture technique, fonctionnelle et organisationnelle du SI
- Connaissance des méthodes, des normes et des outils de développement
- Maîtrise des normes et procédures de sécurité
- Connaissance de l'entreprise, de ses métiers et de son environnement
- Maîtrise de l'anglais technique

Aptitudes professionnelles

- Disponibilité afin de pouvoir faire face à des charges de travail ou à des situations exceptionnelles
- Du sang froid et de la réactivité pour gérer les incidents
- De la rigueur pour appliquer (ou parfois définir) les process
- Bonnes capacités d'analyse pour tirer les bonnes conclusions des rapports d'incidents
- Bonnes aptitudes à la communication pour favoriser les relations avec les autres services et pour dialoguer avec les utilisateurs

- Bonne qualité d'écoute pour prendre en compte les préoccupations et les besoins des utilisateurs
- Capacités pédagogiques pour pouvoir adapter un vocabulaire technique à des non informaticiens

N°7 : Ingénieur développement logiciel

Autres intitulés :

- Ingénieur études et développement
- Ingénieur d'études informatiques
- Ingénieur temps réel
- Développeur informatique embarquée
- Concepteur/Développeur
- Ingénieur test et recette
- Analyste programmeur
- Ingénieur informatique
- Architecte logiciel

Définition :

L'ingénieur développement conçoit, développe et fait évoluer les applications informatiques destinées au système d'information de l'entreprise.

Activités principales :

Traduction technique des besoins fonctionnels

- Participer à l'analyse fonctionnelle détaillée des besoins utilisateurs.
- Etudier les opportunités et la faisabilité technologique de l'application.
- Elaborer et rédiger le cahier des charges techniques, à partir des spécifications fonctionnelles.

Conception et développement

- Concevoir une architecture logicielle avec les architectes, et proposer le framework du projet, constitué de motifs de conception et de librairies.
- Analyser et développer les composants en utilisant les langages appropriés (c++, c, java, c#...).
- Assurer le développement et la réalisation des applications (prototypes et modules).
- Adapter et paramétrer les progiciels retenus pour l'architecture logicielle.
- Participer à la structuration des bases de données.
- Harmoniser et industrialiser l'ensemble des composants et applications.
- Documenter les applications pour les développements ultérieurs et la mise en production.

Tests et recette

- Définir les protocoles et les scenarii de tests (tests unitaires et tests de charge).
- Tester, identifier et traiter les dysfonctionnements éventuels du logiciel développé.
- Analyser les résultats et rédiger le rapport de tests.
- Vérifier la conformité des capacités de l'ouvrage avec la demande formulée par le client.

Mise en production et intégration

- Intégrer les sites pilotes chez le client permettant de tester l'ouvrage, dans sa version définitive.
- Déployer en masse le produit auprès des utilisateurs.

Maintenance évolutive et corrective

La mission de l'ingénieur de développement peut se prolonger après la mise en service des applications. Cette étape porte le nom de tierce maintenance corrective et/ou évolutive applicative (TMA).

Profil :

Diplômes requis

- Formation de niveau bac +2 / 3 (BTS informatique, DUT informatique ou télécommunications, licence professionnelle spécialisée en informatique …)
- Formations professionnelles délivrées par l'AFPA ou le CNAM
- Formation de niveau bac +4/5 (Master) spécialisée en informatique, réseaux et télécommunications
- Écoles d'ingénieurs (informatique, télécoms, généralistes…)
- Une certification professionnelle à certains outils (notamment Microsoft) peut être exigée.

Durée d'expérience

Ce poste s'adresse le plus souvent à des profils débutants ou de jeunes cadres, surtout en informatique de gestion. Il constitue un passage fréquent pour les jeunes diplômés informaticiens.

Il est souvent demandé un stage significatif et/ou deux à trois ans d'expérience professionnelle, en fonction des contraintes technologiques associées au projet.

Certains cadres confirmés peuvent être des experts d'une technologie de développement.

Compétences requises :

Compétences techniques

- Maîtrise des méthodes et outils de développement (UML, Merise, Rational Rose, méthodes agiles, méthodologies objet…)
- Environnement de développement (ex : technologie objet, NETPlateforme de développement : J2EE...)
- Connaissance des applications web (ex : JavaScript, Flash, dreamweaver, Visual Basic, PHP, Flash et autres logiciels associés...)
- Connaissance des framework
- Connaissance de plusieurs langages de programmation (C,C#, javascript, C++, COBOL, Visual C++, Assembleur…), ainsi que de certains systèmes d'exploitation (Windows, Unix, Linux…) ou d'OS (operating system) temps réel (QNX, eCos, VxWorks...)
- Bonne connaissance des normes et procédures de sécurité
- Compréhension de l'environnement et du fonctionnement de l'entreprise

- Une bonne maîtrise de l'anglais technique peut être suffisante mais un bon niveau en anglais est un atout majeur pour comprendre les documentations techniques et/ou se voir confier des missions en environnement international

Aptitudes professionnelles

- Forte motivation pour l'informatique et pour l'apprentissage de nouveaux langages et systèmes
- Souplesse pour répondre aux demandes des clients tout en respectant les normes de développement
- Rapidité d'exécution pour rendre les livrables dans les délais impartis
- Facilité à s'adapter à de nouveaux langages et de nouvelles méthodes de développement
- Polyvalence, créativité pour identifier les solutions techniques appropriées
- Bonnes qualités relationnelles (écoute et expression) permettant de communiquer efficacement avec les utilisateurs
- Ouverture d'esprit, compte tenu de la variété des problématiques clients et technologiques
- Capacité à travailler en équipe
- Autonomie : l'ingénieur développement est souvent amené à exercer son activité en dehors de son entreprise, il lui faut donc faire preuve d'autonomie une fois placé chez un client car il ne dispose pas toujours d'un support
- Adaptabilité en particulier chez les ingénieurs de développement en SSII car les missions sont de durées variables et l'ingénieur peut être affecté chez des clients différents dans un laps de temps limité
- Forte motivation afin de pouvoir s'impliquer dans des projets souvent lourds et longs

N°8 : Ingénieur en informatique industrielle

Autres intitulés :

- Ingénieur automaticien
- Ingénieur CFAO
- Ingénieur GPAO
- Ingénieur GMAO

Définition :

L'ingénieur en informatique industrielle intervient pour optimiser les flux d'information au sein du site de production. Il assure également l'automatisation des tâches industrielles pour améliorer le matériel de l'usine et la productivité des ateliers.

Activités principales :

Analyse du besoin

- Spécifier les besoins de l'utilisateur en termes de fonctionnalités du produit.
- Étudier la faisabilité technologique des applications.
- Élaborer le cahier des charges technique sur la base des spécifications fonctionnelles, de plans, de schémas, de données constructeurs.
- Décomposer le projet en sous-projets spécialisés par modules (interfaces homme/machine, automatisation (PLC), ordre de vision...).

Conception et test de l'outil (réseau, automate, logiciel)

- Concevoir les architectures logicielles, réseaux ou systèmes.
- Programmer et développer les algorithmes de commande (code, bugs, débugs...).
- Réaliser le prototype expérimental et corriger les dysfonctionnements.
- Définir les protocoles et les scénarios de tests (plans de tests).
- Réaliser les tests unitaires et d'intégration de l'outil avant la mise en production.
- Effectuer la validation fonctionnelle de l'outil et rédiger les rapports de test.
- Rédiger le manuel d'utilisation à destination des utilisateurs.

Mise en production et support aux utilisateurs

- Assurer la recette du projet et le suivi de la mise en production de l'outil.
- Conseiller et accompagner techniquement l'utilisateur dans la prise en main de l'outil.
- Prendre en charge les actions de maintenance (curative ou préventive).
- Assurer le service après-vente par des actions de tierce maintenance curative ou évolutive (ajout d'instructions, modification de commandes...).

Veille technologique et industrielle

- Suivre dans la presse spécialisée l'actualité et les progrès en matière d'informatique industrielle (*Mesures*, *Usine nouvelle*...).
- Participer à des salons professionnels et des échanges de bonnes pratiques (à Paris : ESPACE ELEC, MESUREXPO, Salon européen de la recherche et de l'innovation...).

- Suivre le positionnement des concurrents (analyse des plaquettes commerciales, des produits, visites d'entreprises...).

Management de projet

- Évaluer le degré de faisabilité de l'intervention en fonction des compétences disponibles.
- Planifier les délais et fixer le budget nécessaire à l'intervention.
- Veiller à la disponibilité du matériel et affecter les ressources nécessaires (recours à des prestataires externes, évaluation de fournisseurs...).
- Fixer les objectifs des opérateurs et techniciens.
- Suivre l'évolution du projet en fonction des contraintes de coûts, de qualité et de délais et s'assurer de la livraison.
- Piloter les prestataires (SSII...) en cas d'externalisation de tout ou partie de l'informatique industrielle de l'entreprise.
- Évaluer techniquement la prestation des fournisseurs en amont d'un projet ou d'un investissement en lien avec le service achats.

Activités éventuelles

L'ingénieur en informatique industrielle peut également participer aux processus qualité de l'entreprise. Au quotidien, il veille à la qualité du produit/outil qu'il réalise afin de respecter les termes du cahier des charges. Il peut également s'investir dans l'amélioration du management (ISO 9001) en collaboration avec les ingénieurs qualité. Après quelques années d'expérience, l'ingénieur en informatique industrielle prendra en charge le pilotage d'un service dédié. Auquel cas, il sera responsable de l'ensemble des ressources sous sa responsabilité (approvisionnement en fournitures, recrutement, gestion du personnel, gestion de budgets...)

Profil :

Diplômes requis

- Écoles d'ingénieurs spécialisées en génie informatique ou électronique, en informatique industrielle, en automatismes, génie logiciel, en architecture système (Ensam, Esie, ESCPI, Epitech, Efficom…)
- Formation universitaire de type bac + 2 (DUT…) à bac + 5 (DESS/Master) dans les domaines cités ci-dessus

Durée d'expérience

L'accès à des postes d'ingénieur en informatique industrielle peut être ouvert à des jeunes diplômés disposant de stages significatifs. Une expérience de deux ans peut leur permettre de devenir chefs de projets. Enfin, les postes de responsable de service sont confiés à des cadres plus expérimentés disposant d'environ cinq ans d'expérience.

Compétences requises :

Compétences techniques

- Maîtrise des langages de bases de données (SQL, T-SQL...) ou orientés objets (C, C#, java, ADA, Visual Basic.net, Delphi, Labview...)
- Maîtrise des langages de développement d'interfaces web (Html, PHP, ASP...), des réseaux (bus ARINC, AFDX, éthernet/CAN...)
- Maîtrise de l'anglais technique indispensable pour comprendre une documentation ou des interlocuteurs qui s'expriment selon des standards internationaux
- Maîtrise des techniques de base en management afin de pouvoir piloter un projet, voire un service suivant la séniorité du cadre
- Connaissance des méthodes d'analyse d'un projet informatique (UML, Merise, Rational Rose...) et de contrôle qualité (Pareto, Six Sigma, Ishikawa...)
- Bonnes connaissances techniques en mécanique, automatisme, automatique, domotique et électronique...
- Maîtrise de l'anglais et idéalement d'une autre langue étrangère, en particulier lorsque le cadre évolue à l'international

Aptitudes professionnelles

- Écoute et proximité avec le terrain afin d'améliorer la productivité et les conditions de travail en fonction des besoins des utilisateurs
- Ingéniosité afin de trouver des solutions techniques ou organisationnelles : interface homme/machine, logiciel, ou amélioration de postes automatisés
- Mobilité et disponibilité car le poste peut nécessiter de nombreux déplacements ou une forte sollicitation du cadre
- Adaptabilité pour répondre aux besoins des utilisateurs et évoluer chez plusieurs clients
- Rigueur et organisation pour mener à bien un ou plusieurs projets simultanément

N°9 : Chef de projet fonctionnel (web)

Autres intitulés :

- Chef de projet digital
- Chef de projet Internet
- Chef de projet web mobile
- Chef de projet multimédia
- Responsable MOA projet web

Définition :

Le chef de projet fonctionnel web analyse les besoins, organise, planifie et met en œuvre des services ou produits sur Internet pour des clients internes ou externes.

Activités principales :

Analyse des besoins

- Recenser et analyser les besoins des clients (internes ou externes).
- Accompagner les utilisateurs dans leur réflexion sur les usages du Web.
- Participer à la priorisation des projets et à la cohérence des demandes.
- Rédiger les cahiers des charges (s'il est en entreprise) en accord avec les donneurs d'ordre (service marketing, service clients...).
- Rédiger les réponses aux appels d'offres (s'il est prestataire).
- Argumenter les préconisations.
- Définir une enveloppe budgétaire.
- Rédiger l'analyse fonctionnelle détaillée.

Gestion du projet

- Définir l'arborescence des projets.
- Etablir le planning, définir les ressources, le découpage du projet ainsi que le budget détaillé.
- Gérer les priorités du projet.
- Élaborer le concept des projets (définir l'ergonomie, réaliser les story-boards,...)
- Coordonner le travail des différentes équipes : architectes, CP techniques et développeurs (MOE), graphistes, ergonomes, designers, spécialistes gaming, e-commerce, paiement en ligne...
- Suivre l'avancement des éventuels prestataires externes.
- Gérer la recette fonctionnelle (tests utilisateurs).
- Coordonner le déploiement du projet (suivi de la mise en ligne et remontée des incidents ou dysfonctionnements).
- Assurer le suivi de la correction des incidents selon leur nature.

Accompagnement et suivi fonctionnel de la plateforme

- Signaler les « bugs » des fonctionnalités du site et suggérer les évolutions.
- Documenter le projet.- Coordonner avec les équipes techniques et d'exploitation les améliorations à apporter à la plateforme du site Internet (en fonction de l'usage des internautes).

- Assurer une veille fonctionnelle et technique ainsi qu'une veille concurrentielle afin de faire évoluer les fonctionnalités de la plate-forme.
- Accompagner les utilisateurs lors des mises en service.

Activités éventuelles

Peut suivre l'audience du site (en lien avec le *traffic manager* s'il y en a un) ou cumuler la fonction avec celle de *traffic analyser*, voire de référenceur dans les structures de taille petite ou moyenne et notamment analyser les données ou suivre le trafic généré par les internautes.

Dans le cas de projets de petite taille, certains chefs de projets peuvent se voir confier l'ergonomie et/ou l'architecture du site. C'est également ce qui peut se faire dans le cas de projets web mobile pour la réalisation des maquettes et la gestion des problématiques de sécurité et d'ergonomie sur *smartphone*.

Dans le cadre de projets importants, le chef de projet fonctionnel web encadre une équipe de consultants ou d'assistants à maîtrise d'ouvrage qui ont en charge l'écriture des spécifications fonctionnelles.

Profil :

Diplômes requis

- Formation universitaire Bac + 4/5 en informatique complétée par une spécialisation dans le domaine du web (master en stratégie Internet et pilotage des projets d'entreprises, master technologies de l'Internet pour les organisations…)
- Ecole d'ingénieurs ou école de commerce, complétée éventuellement par une spécialisation dans le domaine du Web

Durée d'expérience

Le poste de chef de projet fonctionnel web peut être ouvert à des profils variés :

- Jeunes cadres de 3 à 5 ans d'expérience chez un acteur de l'Internet ou dans une agence web
- Cadres confirmés de 5 ans et plus (avec une expertise confirmée dans la gestion de projets web)

Compétences requises :

Compétences techniques

- Connaître et comprendre le vocabulaire technique utilisé dans les métiers du Net et du multimédia : forum, chat, flux RSS, optimisation des moteurs de recherche, etc.
- Maîtrise de la conduite de projets informatiques en environnement NTIC : techniques de recueil, analyse, qualification des besoins, élaboration des scénarios, de maquettes, rédaction de cahier des charges et chiffrage du projet, dépouillement d'appels d'offre…
- Connaissance des outils et architectures techniques du Web comme JAVA, J2EE, .NET, PHP...
- Bonne culture générale de *webmarketing* (affiliation, partenariats, référencement naturel et payant...)

- La connaissance des outils de e-commerce et/ou de e-CRM peut être demandée pour les personnes qui interviennent sur ce type de projets
- Maîtrise des outils de suivi de projet
- Principes de la qualité, normes régissant les systèmes de qualité et d'ergonomie
- Gestion et contrôle des budgets et des coûts informatiques
- Maîtrise de la modélisation et l'analyse des processus métiers
- Bonne connaissance du domaine fonctionnel dans lequel intervient l'entreprise
- Connaissance des outils de tests

Aptitudes professionnelles

- Diplomatie et écoute car le chef de projet fonctionnel web joue un rôle d'intermédiaire entre les besoins des utilisateurs et les possibilités techniques et budgétaires de la maîtrise d'œuvre et de la production
- Capacité rédactionnelle et sens de la communication car la clarté du cahier des charges et/ou des spécifications fonctionnelles est importante pour la bonne marche du projet
- Bon relationnel car ce cadre est en permanence sollicité, soit par les utilisateurs, soit par les équipes techniques
- Capacité d'adaptation car les évolutions dans ce domaine tant fonctionnelles que techniques sont très rapides
- Goût pour la technique, car il est en contact avec les équipes de maîtrise d'œuvre et de production informatique et, d'autre part, parce que les fonctionnalités prévues dépendent souvent des possibilités données par la technique
- Capacités d'analyse et de synthèse car le chef de projet fonctionnel web a en charge le suivi du projet et doit pouvoir effectuer des reportings sur le respect des coûts et des délais
- Force de proposition pour faire évoluer le contenu (rubriques, thématiques, visuels...), les objectifs (mesure d'audience, publicité on-line...) et la plateforme technique en elle-même (ergonomie, fonctionnalités...)
- Créativité et réactivité, car il doit pouvoir gérer les modifications des projets pour des raisons budgétaires et fonctionnelles ; il doit aussi être capable d'anticiper les évolutions afin de pouvoir intégrer ultérieurement de nouvelles fonctionnalités sans avoir à redévelopper tout le projet

N°10 : Chef de projet maîtrise d'œuvre

Autres intitulés :

- Chef de projet technique
- Responsable technique d'application
- Responsable applicatif
- Chef de projet MOE
- Project leader
- Team leader

Définition :

Le chef de projet spécifie, organise et planifie la mise en œuvre d'un projet, depuis sa phase de conception, jusqu'à son déploiement en s'appuyant sur des ressources internes ou externes.

Activités principales :

Définition de l'ensemble des phases techniques du projet

- Elaborer les spécifications techniques générales du projet sur la base du cahier des charges (fonctionnel) qui a été fourni par la MOA (maîtrise d'ouvrage), en fonction de l'architecture technique et des exigences en termes de sécurité.
- Superviser la rédaction des spécifications (techniques) détaillées du projet.
- Evaluer les risques (coûts, délais…) pouvant intervenir au cours de la réalisation.
- Définir les besoins en termes de ressources humaines et de compétences techniques.
- Préparer en amont les éléments de chiffrage et/ou de facturation.

Pilotage, suivi et coordination du projet

- Mettre en place les structures du projet et ses règles de fonctionnement (méthodes, outils de pilotage, indicateurs…).
- Définir avec les équipes du projet, les objectifs et les délais de réalisation des livrables (applications, modules, développement spécifiques...).
- Choisir et affecter des ressources, en fonction des contraintes techniques du projet.
- Piloter et mesurer l'état d'avancement (création des tableaux de bord, choix des indicateurs, planification des comités de pilotage, …).
- Organiser et animer les comités de pilotage auprès des décideurs.
- Transférer de manière régulière à la maîtrise d'ouvrage les tableaux de bord sur l'état d'avancement du projet.
- Superviser et coordonner le travail de l'ensemble des acteurs internes et/ou externes.
- Valider les livrables.

Test et recette technique

- Planifier et organiser les tests unitaires et de charge
- Suivre la mise en production et le déploiement
- Assurer la correction des anomalies

Profil :

Diplômes requis

- Formation de niveau Bac +2/3 (DUT, BTS, Licence professionnelle) spécialisée en informatique, réseaux et télécommunications
- Formation de niveau Bac +4/5 (Master) spécialisée en informatique, réseaux et télécommunications
- Écoles d'ingénieurs (informatique, télécoms, généralistes..), éventuellement complétées par une formation de type IAE

Durée d'expérience

Le métier de chef de projet informatique est rarement accessible aux débutants. Cependant, certains postes de chef de projet junior peuvent être ouverts à des candidats ayant un à trois ans d'expérience.

Néanmoins de manière générale, le poste de chef de projet suppose une réelle expérience (environ trois à cinq ans) plus ou moins importante selon la taille et la complexité de la mise en œuvre du projet.

Compétences requises :

Compétences techniques

- Bonne connaissance des principaux outils de développement et d'un ensemble de solutions applicatives (ERP, CRM, EAI, …), des bases de données
- Capacité à comprendre les impacts de l'architecture du système d'information sur le projet, et bonne connaissance des architectures applicatives
- Connaissance des processus et méthodes de gestion de projet (planning, budget, indicateurs) et de certains outils de PMO (MS Project)
- Maîtrise des méthodologies et outils de modélisation (UML, Merise...)
- Compréhension de l'environnement et des activités de l'entreprise, des besoins et des contraintes des utilisateurs
- Connaissance dans le domaine de la sécurité des applications informatiques
- Connaissances techniques permettant de venir en appui aux développeurs (selon son domaine de compétences techniques)
- La maîtrise de l'anglais peut être nécessaire pour des projets internationaux ou au sein d'entreprises étrangères

Aptitudes professionnelles

- Bonnes qualités relationnelles et de communication afin d'assurer une collaboration efficace avec son client, et avec les prestataires
- Rigueur et autonomie pour gérer tous les aspects d'un projet (délai, coût, qualité…)
- Esprit d'anticipation de manière à limiter les risques de dérive du projet
- Capacités d'organisation, de planification et de gestion
- Pragmatisme et réactivité pour répondre de manière optimale aux besoins du client
- Qualités d'animateur : écoute, dialogue pour animer et coordonner le travail de son équipe, et comprendre les besoins des utilisateurs finaux

- Capacité à travailler en équipe, car le chef de projet travaille souvent au sein d'équipes importantes
- Qualités d'analyse afin de ne négliger aucun détail, notamment lors des phases de spécifications

N°11 : Chef de projet technique (web)

Autres intitulés :

- Chef de projet Internet technique
- Chef de projet informatique web
- Chef de projet plateforme web mobile
- Lead technique applications Internet/mobile

Définition :

Le chef de projet technique web analyse techniquement les besoins, organise, planifie et met en œuvre le développement informatique des services ou produits sur Internet et/ou web mobile pour des clients internes ou externes. Il doit garantir la qualité technique des applications et veiller au respect des charges et des délais qui ont été impartis.

Activités principales :

Définition de l'ensemble des phases techniques du projet

- Elaborer les spécifications techniques générales du projet sur la base du cahier des charges (fonctionnel) qui a été fourni par la MOA (maîtrise d'ouvrage), en fonction de l'architecture du site web et des exigences de l'expert sécurité web.
- Rédiger les spécifications techniques détaillées du projet.
- Évaluer les risques (techniques, coûts et délais) pouvant intervenir au cours de la réalisation.
- Définir les besoins en termes de ressources humaines et de compétences et constituer les équipes techniques.
- Préparer en amont les éléments de chiffrage et/ou de facturation.
- Assumer le rôle d'expert technique vis-à-vis des concepteurs développeurs

Pilotage, suivi et coordination du projet

- Mettre en place les structures du projet et ses règles de fonctionnement (méthodes, technologies de développement, outils de pilotage, indicateurs...).
- Définir avec les équipes projet, les objectifs et les délais de réalisation des livrables (applications, modules, développements spécifiques...).
- Effectuer les choix et l'affectation des ressources, en fonction des contraintes techniques.
- Piloter et mesurer l'état d'avancement (création des tableaux de bord, choix des indicateurs, planification des comités de pilotage...).
- Superviser et coordonner le travail de l'ensemble des acteurs internes et/ou externes développeurs, designers, graphistes.
- Valider les livrables.
- Planifier et organiser les tests unitaires et de charge.
- Suivre la mise en production et le déploiement.

Gestion de la relation client

- Organiser et animer les comités de pilotage auprès des décideurs.
- Transférer de manière régulière au donneur d'ordre les tableaux de bord sur l'état d'avancement du projet.

- Veiller à maintenir une relation de confiance entre la maîtrise d'ouvrage et les équipes projet.

Profil :

Diplômes requis

- Formation universitaire supérieure informatique Bac + 3/5
- École d'ingénieurs dans le domaine informatique et télécoms
- Ecole d'ingénieurs, complétée par une spécialisation informatique (master communication et technologie numérique, master management de projet web…)

Durée d'expérience

Le poste de chef de projet technique peut être ouvert à des profils variés : jeunes cadres de 2 à 5 ans, ou jeunes diplômés d'écoles d'ingénieurs spécialisées en informatique lorsqu'il s'agit de projets de petite taille.

Les entreprises recherchent également des cadres confirmés (5 ans et plus) avec une expertise dans la gestion de projets techniques complexes et multi plateformes.

Compétences requises :

Compétences techniques

- Connaissances générales en informatique : architecture des systèmes, bases de données, méthodologies de développement, CRM,…
- Excellentes connaissances des langages de programmation spécifiques à l'Internet et à l'Internet mobile (CSS, Java/J2EE, XHTML, AJAX, PHP, ActionScript, *framework*…)
- Connaissance des SGBDR
- Connaissance d'un CMS
- Bonne connaissance de la méthodologie et des outils de tests
- Maîtrise des méthodologies de développement (cycle en V, en escargot, méthode « agile »…) et des méthodes orientées objet

Aptitudes professionnelles

- Rigueur et méthode, car le chef de projet technique doit gérer les plannings, les méthodes, de manière à éviter tout dérapage dans les délais et les budgets
- Goût pour le management d'équipe et la conduite de réunions, car c'est le chef de projet technique qui va avoir en charge l'équipe de développement
- Qualités relationnelles, car il est en relation avec la maîtrise d'ouvrage, la production informatique et la sécurité web et assure l'interface entre les équipes de développement et les équipes artistiques
- Créativité et réactivité, car il doit gérer les modifications des projets pour des raisons budgétaires et fonctionnelles ; il doit aussi anticiper les évolutions futures afin d'intégrer ultérieurement de nouveaux modules sans avoir à redévelopper tout le projet
- Forte motivation pour l'informatique et le développement car il est amené à se plonger dans les problématiques techniques et à faire des choix sur des technologies nouvelles

N°12 : Architecte infrastructures

Autres intitulés :

- Architecte technique
- Architecte système, stockage et réseaux
- Architecte infrastructure globale

Définition :

L'architecte infrastructure garantit la cohérence technique et la pérennité du système d'information lors de ses évolutions tout en veillant à optimiser les ressources, les performances et les coûts.

Activités principales :

Veille et conseil aux équipes de conception

- Conseiller l'urbaniste S.I. sur l'utilisation des outils informatiques et télécoms.
- Pour toute nouvelle technologie, participer aux études d'impact sur l'architecture existante ou prévisionnelle.
- Préconiser des choix techniques afin de garantir la cohérence des évolutions.

Participation à la définition des infrastructures techniques

- Dans le respect des règles d'urbanisme, définir et gérer les standards techniques.
- Assurer la cohérence de l'ensemble des moyens informatiques et télécoms dans le cadre du schéma directeur.
- Réaliser et maintenir la cartographie technique du S.I.
- Définir et faire évoluer le schéma directeur technique.
- Définir et gérer les standards techniques, définir les briques de base du middleware.
- Garantir la cohérence de l'architecture technique avec l'architecture applicative du S.I.
- Identifie les besoins de changements et les composants impliqués : matériels, logiciels, processus, plateforme, en garantissant l'interopérabilité, le dimensionnement, la disponibilité et la sécurité.
- Évalue l'impact des solutions informatique en termes de responsabilités écologiques.

Conduite de projets d'infrastructures

- Analyser les besoins en liaison avec les architectes S.I.
- Analyser l'impact des solutions applicatives retenues en termes d'infrastructures.
- Préconiser des solutions techniques permettant de s'engager sur une qualité et une continuité de service.
- Participation au maquettage de la solution.
- Définir les procédures d'intégration technique.
- Assurer le respect des normes et processus définis dans le cadre du schéma directeur technique.
- Elaborer des procédures de tests permettant d'évaluer la performance, la sécurité, la compatibilité et la fiabilité.

Suivi et amélioration des processus

- Réalise l'audit des infrastructures informatiques de l'entreprise.
- Mesure l'efficacité des processus informatiques en termes d'infrastructures.
- Rédige des recommandations pour l'évolution des solutions informatiques.

Profil :

Diplômes requis

- Formation de niveau Bac +5 (Master 2) spécialisée en informatique et/ou télécoms, sécurité des systèmes informatiques et des réseaux…
- Écoles d'ingénieurs (informatique, télécoms, généralistes..)

Le certificat de qualification professionnelle d'architecte technique (CQP AT) peut être demandé aux personnes travaillant au sein d'entreprises adhérant au SYNTEC.

Durée d'expérience

Ce poste est accessible aux cadres confirmés possédant au minimum 5 ans d'expérience.

Compétences requises :

Compétences techniques

- Bonne connaissance du système d'information global et de l'architecture du SI et des applications
- Excellente maîtrise des systèmes d'exploitation (notamment Windows, UNIX/LINUX), des réseaux et télécoms, des bases de données, stockage (NAS/SAN/DAS)
- Connaissance des technologies et outils de virtualisation (VMware, XEN, RHEV…)
- Compréhension de l'environnement (clients, secteur d'activité, données sensibles…) et du fonctionnement de l'entreprise
- Maîtrise des risques liés à la sécurité des infrastructures et à dématérialisation
- Connaissance des normes dans le domaine de l'archivage et du chiffrement
- Connaissances des problématiques du green IT et du Big Data
- La maîtrise de l'anglais technique est indispensable (documentation en anglais).

Aptitudes professionnelles

- Adaptabilité et curiosité technique car les évolutions technologiques sont rapides et doivent être assimilées afin de pouvoir optimiser l'existant
- Polyvalence, goût pour la technique et créativité pour identifier les solutions techniques appropriées
- Rigueur et esprit critique de manière à faire la part des choses entre la technique et le discours marketing
- Capacité à travailler avec des équipes d'experts
- Une bonne ouverture relationnelle afin de travailler avec d'autres services de l'entreprise (achats, directions métiers notamment)

N°13 : Architecte WEB

Autres intitulés :

- Architecte technique web
- Architecte solutions web
- Architecte J2EE

Définition :

L'architecte web est un expert technique qui a pour principale mission de créer et faire évoluer le schéma technique d'une application ou d'un site web.

Activités principales :

Définition et analyse des besoins clients

- Appréhender les besoins clients (internes ou externes) en matière d'applications web.
- Réaliser l'audit technique des projets notamment sur les problématiques technologiques.
- Collaborer au cahier des charges des projets (création ou évolution de sites web) en listant les fonctionnalités demandées.
- Préparer en amont les éléments de chiffrage et/ou de facturation du projet en termes de matériels ou des sous-traitants (hébergeurs, fournisseurs d'accès…).

Conception de l'architecture et choix technologiques

- Choisir les outils et/ou les infrastructures web.
- Conseiller les clients sur les solutions techniques les plus adaptées à leurs besoins.
- Participer au dimensionnement des projets de manière à garantir la robustesse des solutions techniques mises en œuvre.
- Concevoir et modéliser des architectures, réaliser les maquettes pour présenter les avantages et inconvénients des différentes solutions.
- Evaluer les risques et impacts techniques des solutions préconisées.
- Participer aux calculs de bande passante de manière à optimiser l'architecture d'applications web à fort trafic.
- Dimensionner des serveurs et gérer les relations avec l'hébergeur.
- Participer à la structuration des bases de données (tables, contenus…).
- Concevoir, industrialiser et mutualiser les socles technologiques et couches applicatives (framework, intégration continue, performances, cloud…).
- Suivre les évolutions nécessaires en fonction de l'évolution du nombre de connections ou l'ajout de nouvelles fonctionnalités.

Développement des *framework* et réalisation des tests

- Développer des *framework* (composants logiciels) pour permettre aux développeurs de programmer sur des plateformes telles que J2EE ou .net (*dotnet*).
- Implémenter au sein des équipes les solutions et architectures techniques définies.
- Superviser le bon usage des *framework* lors de la réalisation du codage par les développeurs.
- Vérifier que les fonctionnalités demandées ont été développées correctement grâce aux tests unitaires.
- Mettre en œuvre les tests de charge afin de vérifier la robustesse de l'architecture.

Veille technologique

- Suivre les évolutions technologiques (systèmes, langages, solutions techniques…) et les tester.
- Vérifier la pérennité des solutions existantes.

Profil :

Diplômes requis

- Formations universitaires en informatique Bac + 4/5
- Ecoles d'ingénieurs spécialisées en informatique, télécoms
- Les formations universitaires peuvent être complétées par une certification professionnelle

Durée d'expérience

Le poste d'architecte web est ouvert aux jeunes cadres de 2 à 5 ans d'expérience et aux cadres confirmés (5 ans et plus).

Compétences requises :

Compétences techniques

- Maîtrise des architectures des systèmes web
- Connaissance des langages de programmation spécifiques à l'Internet et éventuellement à l'Internet mobile : Java/J2EE, PHP, .net (*dotnet*), *framework* (Struts, Spring, Hibernate...)
- Connaissance des SGBDR (bases de données)
- Connaissance d'un CMS (Drupal…)
- Bonne connaissance des outils de tests
- Maîtrise de la conduite de projet
- Maîtrise des méthodes de développement (cycle en V, en escargot, méthode agile…) et des méthodes orientées objet
- Notions de gestion des couches télécoms et notamment de la bande passante
- Connaissances hardware pour dimensionner le parc machine nécessaire

Aptitudes professionnelles

- Rigueur et méthode, car il doit prévoir toutes les problématiques techniques auxquelles le site peut être confronté
- Capacité d'abstraction pour être capable de modéliser une solution en prenant en compte l'ensemble des impératifs fonctionnels et technologiques, tout en anticipant les évolutions à venir
- Qualités relationnelles, car il est en relation avec la maîtrise d'ouvrage, la production informatique et la sécurité web et doit pouvoir être l'interface entre les équipes de développement et les équipes d'exploitation et/ou l'hébergeur
- Créativité et réactivité, car il doit pouvoir gérer les modifications des projets pour des raisons budgétaires et fonctionnelles ; il doit aussi être capable d'anticiper les évolutions futures afin de pouvoir intégrer ultérieurement de nouveaux modules sans avoir à redévelopper tout le projet

- Forte motivation pour l'informatique et le développement web car il est amené à se plonger dans les problématiques techniques et à faire des choix sur des technologies nouvelles

N°14 : Responsable d'exploitation

Autres intitulés :

- Responsable technique de compte
- Responsable de production

Définition :

Le responsable d'exploitation informatique supervise l'ensemble de la production informatique de l'entreprise. Il est le garant de son bon fonctionnement, de la mise en place et de la maintenance des applications.

Activités principales :

Définition des procédures d'exploitation

- Définir les procédures et les traitements informatiques afin d'améliorer les performances du système d'information.
- Mettre en place des plans de sauvegarde des données (*back up*).
- Fiabiliser le système notamment par la mise en place de protocoles de sécurité pour assurer un fonctionnement optimal tout en garantissant la sécurité des données.
- Définir la clé de répartition entre externalisation (sous-traitance auprès d'un infogérant) et internalisation.
- Définir les procédures pour assurer une maintenance préventive et curative du système d'exploitation.
- Assurer une veille technologique afin de proposer les évolutions technologiques.

Interface avec les autres services de l'entreprise

- Collecter les besoins des utilisateurs en matière de production informatique.
- Analyser ces besoins sous l'angle technique et rédiger un cahier des charges technique.

Management d'un service d'exploitation

- Définir l'organisation du département et le système d'astreintes.
- Encadrer et animer une équipe composée d'analystes et de techniciens d'exploitation.
- Planifier les interventions et l'allocation des ressources.
- Contribuer au recrutement et prendre en charge tout ou partie du recrutement des nouveaux collaborateurs.

Mesure et contrôle des opérations

- Analyser les tableaux de bord de suivi de l'exploitation.
- Recueillir et analyser les observations et remarques des utilisateurs.
- Analyser les motifs de non-qualité.

Profil :

Diplômes requis

- Diplôme de type Bac +2 en informatique : BTS / DUT informatique
- Diplôme de type Bac +4 en informatique : MIAGE, IUP informatique, maîtrise informatique, ingénieur maître...
- Écoles d'ingénieurs (informatique, télécom, généraliste)
- DESS / DEA informatique

Durée d'expérience

Quatre à cinq ans d'expérience minimum sont nécessaires pour accéder au poste de responsable de l'exploitation.

Compétences requises :

Compétences techniques

- Bonne compréhension du système d'information de l'entreprise
- Maîtrise des différents types de systèmes d'exploitation (grands systèmes, moyens systèmes, micro-informatique) et des serveurs Unix et Windows
- Bonne connaissance des procédures de sauvegarde des données et plus largement des contraintes de sécurité
- Connaissance d'un certain nombre d'outils permettant la sécurisation et la surveillance de l'exploitation tels que Patrol, Asset center…
- Maîtrise de l'anglais technique

Aptitudes professionnelles

- Rigueur et méthode pour organiser de façon optimale les traitements informatiques
- Qualités d'organisation afin de gérer au mieux son service, le budget et les plannings
- Réactivité et disponibilité pour assurer un bon service aux utilisateurs (le responsable est le plus souvent – comme ses collaborateurs – soumis à un système d'astreintes)
- Écoute et dialogue pour comprendre les besoins et les problèmes rencontrés par les utilisateurs
- Sens du service clients/utilisateurs
- Qualités d'animateur afin de motiver ses équipes, les former au process et piloter les projets

N°15 : Responsable de la maîtrise d'ouvrage bancaire/MOA

Autres intitulés :

Consultant manager en maîtrise d'ouvrage bancaire

Définition :

Le responsable de la maîtrise d'ouvrage (MOA) analyse, spécifie et valide fonctionnellement les besoins métiers de la banque. Il pilote les projets d'évolution du système d'information ainsi que les évolutions des processus métiers.

Activités principales :

Analyse des demandes faites par les opérationnels

- Analyser et consolider les demandes de maintenance applicative ou corrective du système d'information métier (suite à des évolutions réglementaires notamment).
- Recueillir et analyser les besoins des équipes métiers pour de nouveaux projets ou des refontes importantes.
- Evaluer le coût des projets.
- Définir les projets prioritaires pour la présentation en comité de projets ou en comité d'investissement (pour les projets nécessitant un budget important) en fonction notamment des obligations réglementaires.
- Présenter les projets en comité d'investissement (ou au directeur de la MOA) en mettant en avant les enjeux, le contexte, les impacts et l'évolution des processus.

Pilotage des pré-études

- Définir les éléments de cadrage du projet : budget, planning, ressources.
- Assurer la mise en place d'une équipe (composée de consultants MOA et d'experts) chargée d'analyser la faisabilité technique des demandes métiers, ainsi que la cohérence avec l'urbanisme et l'architecture globale du SI.
- Animer des réunions avec les responsables des métiers de manière à affiner la demande.
- Identifier les incidences sur les autres applicatifs ainsi que les risques éventuels.
- Chiffrer de manière précise chaque projet et définir le planning et les ressources nécessaires.
- Défendre les projets définitifs et les budgets associés en comité d'investissement.

Gestion des projets et coordination de la mise en œuvre

- Définir les phases du projet et répartir les missions entre les différents consultants ou assistants maîtrise d'ouvrage (internes et/ou prestataires).
- Planifier les réunions et participer au choix des interlocuteurs (spécialistes métiers notamment).
- Suivre les plannings et le budget ainsi que le bon déroulement des différentes phases du projet.
- Animer les comités de pilotage (comprenant notamment des représentants des risques opérationnels, de la conformité, de la sécurité informatique...) et rédiger des comptes rendus.

- Superviser la rédaction du cahier des charges et la validation des spécifications fonctionnelles ainsi que la cartographie des processus.
- Assurer le suivi des livrables auprès de la MOA et gérer la remontée des problèmes rencontrés par les chefs de projets informatiques (ex. : précisions ou incohérences dans certaines spécifications, ou conflits avec d'autres applicatifs).
- Définir le plan de tests et suivre le déroulement des tests fonctionnels utilisateurs.
- Superviser la rédaction de la documentation destinée aux utilisateurs.
- Assurer le reporting du projet auprès du directeur de la MOA et des directeurs des entités concernées par le projet.

Mise en place de la conduite du changement

- Gérer le déploiement des nouveaux applicatifs auprès des utilisateurs.
- Planifier les actions de transfert des compétences aux équipes opérationnelles et organiser des sessions de formation.

Management des équipes de consultants / assistants en maîtrise d'ouvrage

- Assurer le management hiérarchique de son équipe : objectifs, congés, entretiens annuels, besoins de formation...
- Suivre le planning de réalisation des missions des collaborateurs de son équipe affectés à d'autres projets (notamment des projets transverses).
- Suivre le budget alloué.
- Participer au choix et l'évaluation des sous-traitants (sélection des SSII ou cabinets conseil, participation à la rédaction de l'appel d'offres et au dépouillement des réponses, sélection et réception des candidats).

Profil :

Diplômes requis

- Formation de niveau Bac +5 (Master 2) spécialisée en organisation, management de projets, informatique, mathématiques, statistiques, économétrie, banque, audit ou finance
- École supérieure de commerce avec une spécialisation ou un Master 2 en finance
- École d'ingénieurs ou de statistiques avec si possible une spécialisation en finance

Durée d'expérience

Alors que le poste de consultant MOA est proposé prioritairement aux jeunes diplômés et jeunes cadres, le poste de responsable de la maîtrise d'ouvrage est généralement accessible aux cadres confirmés disposant au minimum de 5 ans d'expérience en gestion de projets. En SSII, de jeunes cadres peuvent parfois se voir confier des fonctions de consultant manager.

Compétences requises :

Compétences techniques

- Connaissances générales en informatique : architecture des systèmes, bases de données, méthodologies de développement, CRM, ERP...
- Très bonne connaissance des métiers bancaires
- Maîtrise de l'organisation et de la conduite de réunions
- Maîtrise de l'expression des besoins
- Bonne connaissance de la méthodologie et des outils de tests fonctionnels
- Maîtrise de la rédaction et de la cartographie des processus, ainsi que des outils spécifiques
- Maîtrise des outils bureautiques et de PMO (project management office)
- Connaissance éventuelle d'ERP (enterprise resource planning ou progiciels intégrés), de CRM (customer relationship management ou gestion de la relation client) ou d'outils de BI (business intelligence)
- La maîtrise de l'anglais peut être nécessaire pour certains postes, notamment dans les banques internationales.

Aptitudes professionnelles

- Rigueur et méthode, car le responsable de la MOA doit gérer les plannings, les méthodes, de manière à éviter tout dérapage dans les délais et les budgets
- Goût pour le management d'équipes et la conduite de réunions, car le responsable de la MOA a en charge une équipe de consultants et doit mener fréquemment des réunions avec les représentants des métiers
- Qualités relationnelles, car il est en relation avec les spécialistes métiers, la maîtrise d'œuvre, la production informatique et la sécurité (RSSI) et assure l'interface entre les équipes de développement et les métiers
- Qualités rédactionnelles, pour rédiger les cahiers des charges, les processus (la qualité des applicatifs conçus par la maîtrise d'œuvre informatique dépend pour une grande part de la précision de la formulation rédigée par la MOA)
- Capacité à travailler sur plusieurs projets en parallèle
- Analyse et synthèse, car il doit faire l'arbitrage entre les différents projets en fonction de leur urgence et du retour sur investissement attendu
- Intérêt pour les nouvelles technologies et les problématiques techniques

N°16 : Responsable de parc informatique

Autres intitulés :

- Responsable micro-informatique
- Responsable micro et réseaux
- Gestionnaire de parc informatique et télécoms

Définition :

Le responsable de parc informatique est chargé du bon fonctionnement des postes de travail (fixes et mobiles) et doit garantir la disponibilité permanente aux utilisateurs. Il supervise et assure la maintenance du matériel et l'adapte aux évolutions technologiques ainsi qu'aux besoins de l'entreprise.

Activités principales :

Définition du schéma directeur de l'entreprise en matière de parc informatique

- Conseiller la direction de l'entreprise dans le cadre de l'élaboration du plan d'équipement informatique aux niveaux matériel et logiciel.
- Collecter et analyser les besoins des utilisateurs et participer à l'élaboration des règles d'accès à l'information.
- Définir avec la direction informatique, la politique de maintenance du parc micro (externalisation de tout ou partie).
- Rédiger les cahiers des charges et négocier les contrats avec les fournisseurs et les prestataires de services.

Installation et maintenance du parc informatique

- Assurer le bon fonctionnement du matériel informatique, des périphériques et des logiciels installés.
- Définir et mettre en place l'architecture du réseau local et établir les connexions aux réseaux étendus.
- Garantir la sécurité du matériel et des données sur le réseau local et sur l'ensemble des postes de travail.
- Tester, installer et configurer les nouveaux matériels et suivre leur affectation.
- Définir les procédures de maintenance.
- Assurer la maintenance préventive et curative du parc.

Support et assistance aux utilisateurs

- Assurer le suivi des relations contractuelles et jouer le rôle d'interface entre les prestataires et les clients internes.
- Mettre à la disposition des utilisateurs un support technique afin de collecter et de traiter les problèmes rencontrés.
- Intervenir directement ou envoyer un technicien afin de résoudre le problème de l'utilisateur.
- Organiser des sessions de formation ponctuelles pour les utilisateurs.

Anticipation des besoins et plans d'évolution

- Identifier les besoins à venir et assurer une veille constante sur les nouvelles versions matérielles, logicielles et systèmes.
- Proposer à la direction informatique les moyens logiciels et/ou matériels d'optimiser les coûts et les performances de maintenance.
- Gérer les budgets affectés au renouvellement partiel ou total du parc informatique et à l'achat de nouvelles licences de logiciels.

Pilotage d'un service

- Encadrer et animer une équipe composée d'administrateurs et de techniciens micro et réseaux.
- Mettre en place des normes et procédures (ITIL) pour structurer et optimiser le fonctionnement du service.

Profil :

Diplômes requis

- Diplôme de niveau Bac +2/3 : DUT/BTS, licence pro en informatique
- Diplôme de niveau Bac +4/5 en informatique : IUP informatique, Master informatique…
- Ecole d'ingénieurs (informatique, télécoms, généralistes)

Durée d'expérience

Une expérience minimale de deux à cinq ans dans l'exploitation, la maintenance ou le support technique aux utilisateurs est requise pour accéder à ce poste.

Compétences requises :

Compétences techniques

- Bonne connaissance de l'informatique dans son ensemble
- Bonne connaissance de l'entreprise et de ses métiers pour pouvoir élaborer un plan d'équipement informatique et prévoir les évolutions
- Maîtrise des principaux systèmes d'exploitation du marché : Windows, Linux, Unix…
- Bonne culture générale des grands éditeurs de logiciels du marché et leurs offres : suites logicielles, messagerie, plates-formes collaboratives…
- Maîtrise technique afin de pouvoir déceler et réparer certaines pannes (montage et démontage d'un ordinateur, installation de logiciels en réseau…)
- Bonne connaissance des environnements Windows, réseaux et téléphonie
- L'anglais technique est suffisant pour pouvoir comprendre certains manuels et télécharger des mises à jour sur les sites des constructeurs ou des éditeurs

Aptitudes professionnelles

- Excellentes qualités relationnelles afin de faciliter les contacts quotidiens avec les utilisateurs.
- Pédagogie nécessaire pour former les utilisateurs non informaticiens et éventuellement leur expliquer succinctement les raisons de la panne
- Rapidité et autonomie : le responsable de parc informatique devra parfois intervenir lui-même, il doit donc pouvoir trouver une solution rapide et efficace lorsqu'un incident se produit.
- Disponibilité calquée sur le temps de travail des utilisateurs
- Sens de l'organisation et des priorités pour ne pas s'égarer dans des réflexions ou des actions non prioritaires
- Mobilité car le poste peut exiger des déplacements géographiques

N°17 : Responsable Du SIRH

Autres intitulés :

- Chef de projet SIRH
- Consultant SIRH
- Directeur SIRH

Définition :

Référent sur les systèmes d'information de l'entreprise, le responsable SIRH fait le lien entre les besoins de la DRH en matière de système d'information et la DSI qui gère la maîtrise d'œuvre et permet ainsi l'automatisation de certaines tâches de gestion RH.

Activités principales :

Analyse des besoins de l'entreprise

- Formaliser les besoins de la direction des ressources humaines en matière de système d'information.
- Prioriser les projets en fonction des besoins et de la ligne directrice donnée par la direction générale.
- Cadrer les projets, mener une analyse du retour sur investissement attendu.
- Rédiger les cahiers des charges.

Pilotage du déploiement en interne

- Suivre la mise en œuvre du projet retenu : communication du projet aux équipes RH concernées, animation des réunions de cadrage et du comité de pilotage...
- Mettre en œuvre des plans de test fonctionnels.
- Administrer, paramétrer les systèmes.
- Formaliser les procédures.

Mise en œuvre des évolutions

- Formaliser, centraliser les demandes d'évolution ou de correction émanant des différents utilisateurs au sein du département ressources humaines (la maîtrise d'ouvrage).
- Transmettre à la DSI les dysfonctionnements.
- Proposer des optimisations.

Assistance aux utilisateurs

- Former les utilisateurs du SIRH, notamment les gestionnaires de paie.
- Assurer une assistance auprès de l'ensemble des interlocuteurs RH.
- Rédiger les supports de formation.

Profil :

Diplômes requis

- Formation de niveau Bac +5 (Master) dans deux grandes filières de formation : Master en ressources humaines ou Master en informatique (Master MIAGE [méthodes informatiques appliquées à la gestion des entreprises])
- Écoles d'ingénieurs

Durée d'expérience

Un jeune diplômé peut commencer en tant que chef de projet.

Pour le poste de responsable du SIRH, une expérience de plus de cinq ans est exigée.

Plusieurs années d'expérience dans la gestion d'ERP ou d'outils RH semblent un passage obligé.

Compétences requises :

Compétences techniques

- Parfaite maîtrise des logiciels de paie, de leur paramétrage
- Maîtrise des règles de paie
- Excellentes connaissances en informatique
- Compétences en gestion de projet

Aptitudes professionnelles

- Mobilité dans le cas d'un métier tourné vers l'international
- Organisation pour gérer les projets transverses
- Bon relationnel, capacité à travailler en équipe
- Négociation, diplomatie pour prioriser les projets

N°18 : Responsable informatique

Autres intitulés :

- Responsable du service informatique
- Responsable d'entité
- Responsable informatique PME / filiales

Définition :

Le responsable informatique assure l'organisation, le suivi et la mise en œuvre de toute l'infrastructure système et informatique de l'entreprise.

Activités principales :

Réalisation des objectifs fixés par la DSI

- Participer à la définition de la stratégie et des objectifs en matière de développement informatique.
- Assurer l'organisation, le suivi et la validation des développements informatiques.
- Mettre en place des projets d'évolution en fonction des besoins des utilisateurs.
- Assurer le reporting informatique auprès de la direction.
- Exercer une veille sur les évolutions technologiques et être force de proposition auprès de la direction.

Pilotage du service informatique

- Gérer le budget du service informatique : veiller à la maîtrise des budgets relatifs aux évolutions des systèmes d'information.
- Planifier les activités du service et veiller au respect des plannings.
- Assurer l'encadrement hiérarchique de l'ensemble des équipes informatiques.
- Assurer le pilotage de la sous-traitance : appel d'offres, choix des prestataires, gestion des contrats, suivi technique.

Installation, maintenance et sécurisation du système d'exploitation et d'information

- Définir la politique de maintenance du parc micro.
- Superviser l'achat des équipements informatiques et des logiciels.
- Superviser l'infrastructure des réseaux d'information et garantir leur fonctionnement et leur sécurité.
- Définir les normes et les standards des bases de données, des outils, systèmes ou réseaux.
- Planifier les plans de maintenance.
- Définir les procédures de qualité et de sécurité des systèmes d'information.

Pilotage des projets métiers

- Recenser les besoins des utilisateurs, assurer le suivi et proposer des arbitrages.
- Définir et gérer les ressources humaines et financières.
- Réaliser les tableaux de bord de suivi de l'exploitation
- Garantir le bon respect des cahiers des charges.

Support et assistance aux utilisateurs et fonctions transversales

- Apporter un support technique et une assistance aux utilisateurs.
- Définir l'ensemble des moyens de communication interne nécessaires à la mise en place de nouveaux projets SI.

Profil :

Diplômes requis

- Formation de niveau Bac + 2/3 en informatique avec plusieurs années d'expérience
- Formation de niveau Bac + 4/5 en informatique : IUP informatique, master spécialisé en informatique
- Ecole d'ingénieurs (informatique, télécoms, généraliste)

Durée d'expérience

Selon la taille de l'entreprise et du service informatique, une expérience de cinq ans dans une SSII ou dans une entreprise utilisatrice est nécessaire.

Compétences requises :

Compétences techniques

- Connaissance large des systèmes d'information
- Connaissance des applications et des technologies utilisées dans l'entreprise, des principaux langages informatiques et systèmes d'exploitation
- Maîtrise des normes de sécurité et de l'actualité des risques mondiaux en matière de sécurité
- Bonne connaissance du marché de la sous-traitance : éditeurs, SSII, cabinets de conseil et gestion de la relation avec la sous-traitance
- Connaissance des métiers et de l'organisation de l'entreprise, des besoins des autres directions
- Certaines bases en contrôle de gestion, pour piloter la gestion du budget informatique et favoriser le dialogue avec la direction financière
- Bonne maîtrise des méthodologies de gestion multi projets
- Maîtrise de l'anglais technique

Aptitudes professionnelles

- Sens de l'anticipation pour mettre en œuvre des solutions innovantes

- Adaptabilité car le secteur informatique évolue toujours et une veille technologique est primordiale pour maintenir le système d'information en état de fonctionnement et l'optimiser
- Qualités relationnelles et sens de l'écoute dans ses rapports fonctionnels et hiérarchiques
- Capacité à négocier avec les collaborateurs (définir les objectifs, participer au processus de recrutement) et les prestataires (obtenir le produit ou le service offrant le meilleur rapport qualité/prix pour l'entreprise)
- Bonnes compétences rédactionnelles pour la formalisation de clauses contractuelles et la présentation à la direction générale des évolutions nécessaires
- Pédagogie car le responsable informatique doit être capable d'expliquer les changements liés à de nouveaux projets informatiques
- Bonne expression écrite et orale

N°19 : Analyste d'exploitation

Autres intitulés :

- Ingénieur d'exploitation
- Ingénieur de production
- Analyste de production
- Analyste de réseaux
- Analyste métrologie
- Analyste support diffusion maintenance

Définition :

L'analyste d'exploitation met en place et garantit le fonctionnement technique des moyens de production et des systèmes d'information et contribue à leur optimisation.

Activités principales :

Définition des procédures d'exploitation

- Participer à l'étude et à l'analyse des besoins de l'entreprise en matière de système d'exploitation.
- Élaborer et diffuser des procédures d'exploitation.
- Mettre en place des tableaux de bord de suivi des performances

Réalisation de l'intégration technique d'applications

- Installer les logiciels et les matériels retenus.
- Configurer les postes de travail.
- Automatiser la production et l'exploitation.
- Planifier les traitements et les sauvegardes.
- Assurer l'évolution et la maintenance.
- Assurer une surveillance en temps réel des systèmes et des applications.
- Mettre en place un programme de tests.

Support aux utilisateurs

- Assurer un support technique aux utilisateurs.
- Intervenir rapidement lors des incidents d'exploitation (analyse, diagnostic et résolution).

Analyse des rapports techniques

- Collaborer à la définition ou à l'amélioration des procédures sécurité.
- Établir la documentation technique.
- Veiller à l'efficacité permanente des systèmes de sécurité.
- Suivre les incidents d'exploitation, en analyser les causes et prendre les mesures correctives associées.

Profil :

Diplômes requis

- Diplôme de type Bac +2 en informatique : BTS/DUT informatique
- Diplôme de type Bac +4 en informatique : MIAGE, IUP informatique, maîtrise informatique, ingénieur maître...
- Écoles d'ingénieurs (informatique, télécoms, généraliste)
- DESS/DEA informatique

Durée d'expérience

Ce poste peut être proposé à des débutants, mais s'adresse généralement à des cadres ayant de deux à trois ans d'expérience professionnelle.

Compétences requises :

Compétences techniques

- Connaissance des matériels, des logiciels, des normes de fichiers, des systèmes d'exploitation, des applications et des liaisons inter applications.
- Connaissance des procédures de transmission de données.
- Bonne connaissance des systèmes d'exploitation et en particulier des serveurs Unix (type Solaris) et des bases de données associées (Oracle).
- Très bonne connaissance des environnements Microsoft (Windows NT, Microsoft Exchange).
- Bonne appréhension du monde de l'open source (Linux).
- Maîtrise de l'écriture de script (type Schell).
- Connaissance des ordonnanceurs type $ Universe.
- Connaissance de l'architecture fonctionnelle et organisationnelle du SI.
- Connaissance de l'entreprise, de ses métiers et de son environnement.
- Maîtrise de l'anglais technique.

Aptitudes professionnelles

- Disponibilité afin de pouvoir faire face à des charges de travail ou à des situations exceptionnelles.
- Du sang-froid et de la réactivité pour gérer les incidents.
- De la rigueur pour appliquer et parfois définir les process.
- Capacités d'analyse pour tirer les conclusions des rapports d'incidents.
- Aptitudes à la communication pour favoriser les relations avec les autres services et pour dialoguer avec les utilisateurs.
- Bonnes qualités d'écoute pour prendre en compte les préoccupations et les besoins des utilisateurs.
- Capacités pédagogiques pour pouvoir adapter un vocabulaire technique à des non-informaticiens.

N°20 : Administrateur de bases de données

Autres intitulés :

- Database administrator (DBA)
- Ingénieur en bases de données
- Responsable de bases de données
- Ingénieur datawarehouse

Définition :

L'administrateur base de données conçoit, gère et administre les systèmes de gestion de bases de données ; il garantit la cohérence, la qualité, la sécurité et l'accessibilité permanente des informations.

Activités principales :

Conception de bases de données

- En étroite collaboration avec l'architecte S.I., mettre en place des standards en ce qui concerne les bases de données, préconiser des bonnes pratiques à usage des équipes de développement.
- Effectuer les choix d'implémentation des bases de données dans le respect du cahier des charges et en collaboration avec les différents acteurs du projet (chef de projet, architectes, intégrateurs, consultants ERP, experts informatique décisionnelle…).
- Définir de manière optimale les paramètres de la base de données.
- Définir des règles de sécurité (physique et logique) des données ainsi que des normes d'utilisation des bases.
- Modéliser la base et concevoir les tables et les clefs.
- Prendre en compte les spécificités du client interne ou externe en collaboration avec l'architecte infrastructure en ce qui concerne notamment la taille de la base (capacity planning).

Administration et maintenance des bases de données

- Créer les bases et implanter les données sur les supports physiques.
- Dimensionner le serveur.
- Garantir la disponibilité et la qualité des données par le maintien et l'amélioration des performances (« tuning ») et fonctionnalités (en améliorant leur automatisation, optimisant les traitements et les requêtes, les paramétrages…).
- Suivre les statistiques des performances d'accès aux objets de la base de manière à assurer ainsi le tuning.
- Gestion des montées en charge (suivi de la volumétrie…) sur l'aspect accès aux données.
- Administrer les autorisations d'accès pour les utilisateurs ainsi que les problématiques de sécurité des données avec l'ingénieur sécurité.
- Gérer des migrations de version.
- Mettre à jour les programmes et corriger ses éventuels bugs (passer des patches) et donner son aval avant les mises en production.

- Préconiser des dispositifs de sauvegarde (back up) à réaliser par les services d'exploitation pour assurer l'intégrité de l'ensemble des données de la base, notamment, l'archivage et la purge des données.
- Définir des normes qualité et élaborer des tableaux de bord et pour en assurer le suivi.
- Assurer le suivi des outils de supervision et étudier l'ensemble des incidents survenus afin de comprendre leur gravité et leurs origines.
- Elaborer et exécuter les procédures et programmes de test, lors des migrations ou à la suite d'un incident.
- S'assurer que les sauvegardes sont bien effectuées (en allant faire des tests dans un centre de back-up) et contrôler les mouvements sur les données.
- Assurer la récupération des données et la remise en condition opérationnelle des bases de données (disaster recovery) suite à incident grave.

Support technique et assistance aux informaticiens et aux utilisateurs

- Assurer le support aux développeurs et aux techniciens d'exploitation.
- Assister la MOA sur le plan technique en développant leur maîtrise des outils.
- Mettre un support technique de niveau 2 à la disposition des utilisateurs et définir les procédures d'intervention afin de résoudre les éventuels problèmes.
- Intervenir immédiatement en cas d'incidents limitant les performances des bases de données ou l'accès aux informations.

Veille technologique et contrôle de la base de données

- Assurer une veille technologique afin de maintenir l'adéquation des performances et des spécificités des bases de données aux besoins de l'entreprise.
- Suivre et contrôler les évolutions de version des bases existantes pour faire évoluer les bases de données
- Tester et valider les systèmes de gestion de bases de données dans le cadre de migrations ou d'évolution technologique.
- Définir les normes et standards d'utilisation et d'exploitation des systèmes de gestion de bases de données
- Assurer la mise à jour de la documentation sur la structure de la base, et les procédures d'exploitation et de production.
- Présenter de nouveaux produits ou de nouvelles versions à son entreprise.

Profil :

Diplômes requis

- Formation de niveau Bac +5 (Master 2) spécialisée en informatique et/ou télécoms, sécurité des systèmes informatiques et des réseaux, …
- Écoles d'ingénieurs (informatique, télécoms, généralistes...)
- La certification de qualification professionnelle (CQP) administrateur de base de données peut être demandée aux personnes travaillant dans des entreprises adhérant au SYNTEC

NB : Une certification délivrée par les éditeurs (type Oracle) peut être requise par certaines entreprises.

Durée d'expérience

Ce poste s'adresse plutôt à des jeunes cadres disposant de 2 à 3 ans d'expérience professionnelle et à des cadres confirmés.

Compétences requises :

Compétences techniques

- Bonne connaissance du système d'information global et de l'architecture du SI et des applications
- Compétences en systèmes et réseaux
- Maîtrise des bases de données (Oracle, MySQL, SyBase, SQL Server…) et des outils spécialisés dans l'administration de ces bases
- Maîtrise du langage de requête SQL
- Connaissance des scripts Shell sous UNIX, Windows ou MVS
- Compréhension de l'environnement (clients, secteur d'activité, données sensibles…) et du fonctionnement de l'entreprise
- Maîtrise des risques liés à la sécurité
- La maîtrise de l'anglais technique est indispensable (documentation en anglais)

Aptitudes professionnelles

- Adaptabilité et curiosité technique car les évolutions technologiques sont rapides et doivent être assimilées afin de pouvoir optimiser l'existant
- Réactivité, et bonne gestion du stress pour intervenir rapidement mais en tenant compte des problématiques de sécurité en cas d'incidents
- Rigueur, sens de la méthode et précision car les bases de données sont un outil stratégique pour de nombreuses entreprises
- Bonne appréhension du risque, aussi bien technique que lié au contenu des données
- Bonne résistance à la répétition des tâches, notamment par rapport aux activités de traitements des incidents, de suivi des flux
- Faculté à synthétiser les besoins des informaticiens ou utilisateurs afin d'y répondre au mieux

N°21 : Administrateur réseau

Autres intitulés :

Administrateur réseau et système

Définition :

L'administrateur réseau est garant du bon fonctionnement et de la qualité du réseau de l'entreprise. Il participe à son évolution et pilote l'accès aux utilisateurs.

Activités principales :

Gestion de projets et mise en place du réseau

- Optimiser le réseau par la conduite de projet d'installation ou de refonte de certains éléments du réseau de l'entreprise, matériels ou logiciels.
- Prendre en compte les exigences des utilisateurs en termes d'exigence de performances du réseau (puissance, rapidité, stabilité).
- Intégrer de nouvelles applications afin d'améliorer les performances des réseaux.
- Assurer l'interface entre les équipes internes et externes (sous-traitants) lors de la mise en place de réseaux.
- Mettre en place les interconnexions entre les différents réseaux de l'entreprise pour assurer leur compatibilité.
- Apporter son expertise technique et fonctionnelle sur la partie réseaux lors du lancement de projets transverses.

Administration des réseaux

- Mettre en place les normes de sécurité, notamment celles liées aux conditions d'accès.
- Assurer la bonne gestion des droits d'accès, pour les machines d'une part, et pour les utilisateurs d'autre part, dans le respect des règles de sécurité de l'entreprise.
- Mettre en place des tableaux de bord de suivi des performances et de qualité du réseau (pannes, flux, disponibilité des ressources, sécurité, etc.).
- Installer les logiciels d'administration de réseau.
- Assurer l'ensemble des sauvegardes nécessaires pour maintenir la sécurité des données circulant dans le réseau de l'entreprise.
- Suivre le budget d'exploitation des réseaux.

Support aux utilisateurs et maintenance réseau

- Assister les utilisateurs (hotline) sur la partie réseau afin de les aider en cas de panne ou de difficultés.
- Diagnostiquer, prévenir et réparer les pannes et les dysfonctionnements des réseaux.
- Former et sensibiliser les utilisateurs aux réseaux et à la sécurité.

Veille technologique

- Assurer une veille technologique afin d'anticiper les évolutions nécessaires à l'optimisation du réseau.
- Proposer les investissements informatiques relatifs au réseau.

Profil :

Diplômes requis

- Écoles d'ingénieurs spécialisées en informatique, réseau ou télécommunications
- DESS/DEA spécialisés en administration système et réseau
- DESS/DEA informatique
- Diplôme de type Bac +4 en informatique : MIAGE, IUP informatique, maîtrise informatique, ingénieur maître...
- Diplôme de type Bac +2 en informatique : BTS/DUT informatique

Durée d'expérience

Le poste d'administrateur réseau peut être accessible aux jeunes diplômés. Il s'adresse toutefois prioritairement aux cadres disposant de un à trois ans d'expérience dans les domaines de l'administration réseau ou du support aux utilisateurs

Compétences requises :

Compétences techniques

- Expertise dans l'administration des réseaux et systèmes
- Bonnes connaissances de l'architecture et des fonctionnalités du SI de l'entreprise
- Connaissances des protocoles de communication
- Maîtrise des normes et procédures de sécurité informatique et télécommunications
- Bonne connaissance en technologies télécoms, Internet (Web, XML, PHP…), ainsi qu'en bases de données (Oracle, SQL Server...)
- Très bonnes connaissances des principaux systèmes d'exploitation (Windows et Unix)
- Compréhension de l'environnement de l'entreprise et de ses spécificités métiers
- Maîtrise de l'anglais technique

Aptitudes professionnelles

- Sens de l'écoute et du dialogue pour bien comprendre les besoins des utilisateurs
- Rigueur et sens de la méthode car l'administration d'un réseau de plusieurs centaines de postes requiert une attention de tous les instants
- Bonne appréhension du risque, aussi bien technique que lié au contenu des données
- Résistance au stress afin de faire face à la pression en cas de panne
- Réactivité et disponibilité pour assurer un service performant aux utilisateurs
- Patience, soit dans le cadre d'un diagnostic particulièrement difficile, soit dans le cadre de la formation des utilisateurs aux règles de sécurité réseau
- Adaptabilité et curiosité car les évolutions technologiques sont rapides et doivent être assimilées afin de pouvoir optimiser l'existant

N°22 : Ingénieur système

Autres intitulés :

- Ingénieur systèmes réseaux
- Responsable systèmes

Définition :

L'ingénieur systèmes réseaux est responsable de la mise en place et de la maintenance des matériels et logiciels liés aux systèmes d'exploitation.

Activités principales :

Analyse des besoins et veille technologique

- Recueillir l'information nécessaire et étudier les besoins d'équipements matériels et logiciels.
- Préconiser les solutions informatiques en réponse aux besoins matériels et conseiller sur les choix finaux.
- Rédiger le cahier des charges contenant les spécifications techniques des équipements.
- Rédiger les appels d'offre et analyser les propositions des constructeurs en termes de performance, fiabilité et compatibilité.
- Assurer une veille technologique pour garantir l'optimisation des ressources systèmes de l'entreprise.

Mise en place et intégration des systèmes d'exploitation retenus

- Conseiller les équipes de la DSI sur l'utilisation des ressources du système, des langages, des fichiers…
- Concevoir ou adapter (paramétrer) les logiciels de base sélectionnés.
- Configurer et dimensionner les solutions hardware retenues en fonction des performances requises par les logiciels.
- Tester les systèmes mis en place et veiller à la compatibilité des éléments entre eux,
- Rédiger et mettre à jour la documentation des procédures et consignes d'exploitation.
- Veiller à la sécurité et à la fiabilité des systèmes de l'entreprise.
- Participer aux phases de validation technique lors des mises en production.

Maintenance du système d'exploitation

- Gérer les ressources systèmes et les comptes utilisateurs
- Diagnostiquer les pannes et les dysfonctionnements liés au hardware ou aux logiciels.
- Réparer les pannes et les dysfonctionnements.
- Réaliser les installations from stratch des serveurs
- Assurer une maintenance évolutive et corrective en fonction des grandes évolutions technologiques (notamment les changements de versions)
- Mesurer et optimiser les performances des systèmes d'exploitation (tuning).

Profil :

Diplômes requis

- Formation de niveau Bac +2/3 (DUT, BTS, Licence professionnelle) spécialisée en informatique, réseaux et télécommunications
- Formation de niveau Bac +5 (Master) spécialisée en informatique, réseaux et télécommunications
- Écoles d'ingénieurs (informatique, télécoms, généralistes...)

Durée d'expérience

Le poste d'ingénieur système peut être un premier emploi pour les ingénieurs.
Cette fonction s'adresse prioritairement aux cadres disposant de 2 à 5 ans d'expérience dans l'administration des systèmes ou la production informatique.

Compétences requises :

Compétences techniques

- Maîtrise voire expertise des logiciels de l'infrastructure technique notamment des systèmes d'exploitation (UNIX, LINUX, MVS, Windows…), et des interprétateurs de commandes (Shell) pour diagnostiquer et réparer les dysfonctionnements
- Bonne maîtrise des technologies Internet : protocoles de sécurité, protocoles Internet…
- Bonne connaissance de l'architecture et des fonctionnalités du SI de l'entreprise
- Connaissance des principaux constructeurs et éditeurs prestataires du marché et notamment de l'offre hardware du marché
- Connaissance des normes ITIL ou ISO (si l'entreprise dispose d'une certification)
- Connaissance des bases de données (Oracle, SQLServer…) et des serveurs de messagerie
- Connaissance des normes et procédures de sécurité et des outils associés
- Bonne culture informatique : principaux langages et outils de développement de l'entreprise
- Bonne connaissance des clients internes de la DSI : les principaux métiers de l'entreprise
- Maîtrise de l'anglais technique

Aptitudes professionnelles

- Rigueur, organisation et méthode : les tâches de l'ingénieur systèmes requièrent le respect des méthodes et des règles de qualité définies en interne
- Capacité d'analyse pour faire en sorte que, confronté à une situation d'urgence, l'ingénieur systèmes trouve la solution la plus adaptée
- Résistance au stress afin de résister à la pression en cas de panne
- Bonnes qualités relationnelles dans les rapports avec les utilisateurs internes et externes
- Réactivité et disponibilité : la mission de maintenance de l'ingénieur systèmes implique de sa part la capacité à réagir rapidement en cas d'incident ou de panne
- Capacité d'anticipation et de prises d'initiative pour assurer une veille technologique efficace
- Adaptabilité et curiosité technique car les évolutions technologiques sont rapides et doivent être assimilées afin de pouvoir optimiser l'existant
- Polyvalence, car il doit assister dans de nombreuses entreprises les administrateurs réseaux et les ingénieurs télécoms

N°23 : Ingénieur sécurité WEB

Autres intitulés :

- Expert sécurité des SI
- Consultant sécurité des applications web
- Ingénieur/responsable sécurité informatique
- Auditeur sécurité informatique

Définition :

L'ingénieur sécurité web contribue à la mise en œuvre de la politique de sécurité du système d'information notamment pour tout ce qui concerne les flux avec l'extérieur de l'entreprise (site, messagerie, paiement, identification,…) Il est chargé d'évaluer la vulnérabilité du système d'information de l'entreprise, de proposer au RSSI des solutions pour développer la politique de sécurité et d'installer des procédures de protection des réseaux informatiques contre toute intrusion extérieure (virus, hackers...).

Activités principales :

Analyse des risques, études et audit de la sécurité web

- Auditer le système de sécurité web, wifi, VoIP, éventuellement avec l'aide de prestataires (tests de pénétration et d'intrusion).
- Analyser les risques, les dysfonctionnements, les failles dans la protection, les marges d'amélioration des systèmes de sécurité.
- Définir ou faire évoluer les mesures et les normes de sécurité web et messagerie, en cohérence avec la nature de l'activité de l'entreprise et son exposition aux risques informatiques (politique de mots de passe, choix d'antivirus, certificats…).
- Réaliser les études techniques permettant au RSSI de faire les choix des dispositifs techniques les plus appropriés aux besoins de l'entreprise (*firewall*, cryptographie, authentification...).

Mise en œuvre et suivi du dispositif de sécurité Internet

- Mettre en place les méthodes et outils de sécurité web adaptés et accompagner leur implémentation auprès des utilisateurs.
- Élaborer et suivre les tableaux de bord des incidents de sécurité Internet (attaques virales notamment).
- Réparer les dommages causés au SI en cas d'intrusion dans le système ou de contamination par un virus, en analyser les causes et consolider les mesures de sécurité.
- Tester ou faire tester régulièrement le bon fonctionnement des mesures de sécurité mises en place pour en détecter les faiblesses et les carences (tests d'intrusion notamment).

Communication et formation sur les normes de sécurité

- Participer à la réalisation du référentiel de sécurité, sur la partie sécurité des réseaux (politique de mots de passe, d'authentification, d'utilisation de certificats, de niveau de sécurité antivirale sur les postes, de définition (censoriale) de sites de confiance…) l'actualiser régulièrement, en assurer la diffusion auprès des utilisateurs et veiller à son application.

- Réaliser des supports de formation et en assurer la diffusion principalement auprès des collègues du service informatique.
- Mettre en place des actions de communication auprès des salariés de l'entreprise en cas de risque majeur (information sur des types de mails infectés par exemple) ou de dommages au SI causés par une attaque.

Veille technologique et réglementaire

- Assurer une veille technologique, notamment sur les protocoles, les nouveaux systèmes d'intrusion et les dernières techniques d'attaque sur le web ainsi que les évolutions des protections pour garantir la sécurité du système.
- Identifier les nouveaux risques sur la sécurité du système d'information : apparition de nouveaux virus, lancement d'attaques informatiques sur le réseau mondial...
- Suivre les évolutions juridiques du marché en termes de sécurité Internet afin de garantir que les mesures de sécurité web soient bien conformes au droit individuel et collectif.

Profil :

Diplômes requis

- Écoles d'ingénieurs (informatique, télécoms, généralistes…)
- Masters spécialisés en sécurité informatique et/ou télécoms, sécurité des systèmes informatiques et des réseaux, sécurité, cryptologie et codage de l'information…

Durée d'expérience

Selon le niveau d'expertise requis, le poste d'ingénieur sécurité web est ouvert à des personnes confirmées ou à de jeunes cadres ou de jeunes diplômés possédant des profils techniques pointus dans le domaine réseau télécoms (principalement en société de conseil).

Il faut également noter que ce poste a pu parfois être ouvert (chez certains éditeurs) à d'anciens *hackers* repentis quel que soit leur diplôme.

Compétences requises :

Compétences techniques

- Bonne connaissance de la stratégie de l'entreprise, de son organisation, de ses métiers et des enjeux
- Bonne connaissance du système d'information global, de l'urbanisation et de l'architecture du SI et des interfaces en applications
- Maîtrise des normes et procédures de sécurité et des outils et technologies qui s'y rapportent : *firewall*, antivirus, cryptographie, serveurs d'authentification, tests d'intrusion, PKI, filtrages d'URL...
- Bonne connaissance des systèmes d'exploitation (MVS, UNIX, Linux, Windows…) et des langages de programmation associés
- Bonne connaissance des outils d'évaluation et de maîtrise des risques (méthode Marion), des réseaux et systèmes
- Connaissance des méthodologies (ex : OSSTMM, OWASP…)
- Connaissance des principaux prestataires du marché de la sécurité informatique (éditeurs, sociétés de service...)

- Bonnes connaissances juridiques en matière de sécurité et de droit informatique
- Maîtrise de l'anglais, car 90 % des documents relatifs à la sécurité sont rédigés en anglais

Aptitudes professionnelles

- Curiosité et goût pour la technique, car l'expert sécurité doit être au courant en permanence des nouveaux risques et des nouvelles parades (virus et antidotes)
- Diplomatie, écoute sens du dialogue, persuasion, pour convaincre les utilisateurs des risques encourus et du bien-fondé des procédures mises en place
- Intégrité et éthique, car il a accès à toutes les données sensibles de l'entreprise et il doit maintenir un bon niveau de confidentialité
- Capacité à gérer le stress dans des situations de crise (intrusion, virus, problème de sécurité « matérielle » (incendies, fuites d'eau…) et à prioriser les actions à mener
- Rigueur, capacité d'anticipation et méthode afin de pouvoir prévoir les actions à mener en cas de crise
- Capacités d'analyse afin de planifier minutieusement les risques et leurs parades
- Force de proposition pour faire évoluer la stratégie, ainsi que les pratiques
- Capacités pédagogiques pour vulgariser les risques et les enjeux de la sécurité à tous les niveaux dans l'entreprise
- Capacité à travailler avec tous les niveaux d'interlocuteurs de l'entreprise en adaptant son langage et son niveau d'explication à la population avec laquelle l'ingénieur sécurité web est amené à travailler

N°24 : Responsable sécurité informatique

Autres intitulés :

- Responsable ou directeur de la sécurité des systèmes d'information (RSSI ou DSSI)
- Responsable des risques de la sécurité des systèmes informatiques (RRSSI)
- Responsable sécurité des réseaux informatiques

Définition :

Le responsable sécurité informatique évalue la vulnérabilité du système d'information de l'entreprise, définit et met en œuvre la politique de sécurité de l'entreprise. Il met en place des solutions pour garantir la disponibilité, la sécurité et l'intégrité de système d'information et des données.

Activités principales :

Identification des risques et définition de la politique de sécurité

- Réaliser des audits du système de sécurité, le plus souvent avec l'aide de prestataires.
- Analyser les risques et les dysfonctionnements, les marges d'amélioration des systèmes de sécurité.
- Définir et faire évoluer la politique de sécurité des systèmes d'information du Groupe (PSSI).
- Etablir un plan de prévention des risques informatiques et un plan de continuité d'activité (PCA) (ou plan de maintien en conditions opérationnelles du S.I.).
- Définir ou faire évoluer les mesures et les normes de sécurité (charte), en cohérence avec la nature de l'activité de l'entreprise et son exposition aux risques informatiques (nomadisme, BYOD (Bring your own device), transferts de données, transactions financières…).
- Choisir les dispositifs techniques les plus appropriés aux besoins de l'entreprise (firewall, programmes de back up, cryptographie, authentification…).
- Participer à la définition et au contrôle de la gestion des habilitations.
- Participer au comité des risques.

Mise en œuvre et suivi du dispositif de sécurité

- Faire appliquer les normes et standards de sécurité.
- Mettre en place les méthodes et outils de sécurité adaptés, et accompagner leur implémentation auprès des utilisateurs.
- Gérer les projets d'infrastructures sécuritaires ;
- Elaborer et suivre des tableaux de bord des incidents sécurité.
- Superviser ou auditer les programmes de sauvegarde (back-up).
- Gérer les incidents sécurité et proposer des solutions pour rétablir rapidement les services.
- Définir les actions à mener afin de réparer les dommages causés au SI en cas de survenance d'un sinistre de sécurité S.I. (intrusion dans le système, contamination par un virus, défaillance d'un équipement…), mettre en œuvre le plan de reprise d'activité (PRA).
- Faire analyser les causes des incidents et consolider les mesures de sécurité.
- Faire tester régulièrement le bon fonctionnement des mesures de sécurité mises en place pour en détecter les faiblesses et les carences.
- Auditer le respect des normes de sécurité informatique imposées aux sous-traitants de l'entreprise.

Communication et formation sur les normes de sécurité

- Réaliser le référentiel de sécurité, l'actualiser régulièrement, en assurer la diffusion et veiller à son application.
- Définir les formations à réaliser, superviser la rédaction des supports de formation et en assurer la diffusion (principalement auprès du service informatique).
- Mettre en place des actions de communication (en concertation avec le responsable de l'exploitation informatique ou les risk managers métiers) auprès des salariés de l'entreprise en cas de risque majeur ou de dommages au SI causés par une attaque ou par des dégâts matériels.

Veille technologique et réglementaire

- Assurer une veille technologique, de manière à garantir la sécurité logique et physique du système d'information.
- Assurer une veille réglementaire sur la protection des données personnelles.
- Identifier les nouveaux risques sur la sécurité du système d'information : apparition de nouveaux virus, lancement d'attaques informatiques sur le réseau mondial...
- Rechercher des solutions innovantes pour répondre aux problématiques induites par l'introduction d'une nouvelle technologie.
- Suivre les évolutions juridiques du marché en termes de sécurité informatique afin de garantir la conformité du SI au droit individuel et collectif.
- Rédiger des notes technologiques de sécurité.

Management des équipes de correspondants / ingénieurs sécurité

- Assurer le management hiérarchique de son équipe : objectifs, congés, entretiens annuels, besoins de formation…
- Définir les plannings ainsi que la participation à des projets transverses (notamment comités risques dans la banque).
- Suivre l'action des correspondants sécurité.
- Suivre le budget alloué à la sécurité informatique.
- Participer au choix et l'évaluation des sous-traitants (sélection des SSII ou cabinets conseil, participation à la rédaction de l'appel d'offre et au dépouillement des réponses, sélection et réception des candidats).

Suivi des actions et reporting

- Contrôler les tableaux de bord techniques des incidents de sécurité rencontrés (virus, tentatives d'intrusion, …)
- Assurer le reporting des problèmes de sécurité en estimant les pertes financières (pertes engendrées et coût de mise en place d'une parade).

Profil :

En entreprise :

- Directeur du système informatique (DSI) ou Directeur du système d'information et de l'organisation (DSIO)
- Directeur technique
- Directeur de la production informatique ou des infrastructures informatiques

En SSII ou société de conseil :

- Directeur de mission
- Directeur technique

Compétences requises :

Compétences techniques

- Bonne connaissance de la stratégie de l'entreprise, de son organisation, de ses métiers et des enjeux
- Bonne connaissance du système d'information global, de l'urbanisation et de l'architecture du SI et des interfaces en applications
- Maîtrise des normes et procédures de sécurité et des outils et technologies qui s'y rapportent: firewall, antivirus, cryptographie, serveurs d'authentification, tests d'intrusion, PKI, filtrages d'URL...
- Connaissance des principaux prestataires du marché de la sécurité informatique (éditeurs, sociétés de service...)
- Bonne connaissance des réseaux et systèmes
- Bonne connaissance des outils d'évaluation et de maîtrise des risques (méthode Marion MEHARI…)
- Connaissance des méthodologies (ex : OSSTMM, OWASP…)
- Bonnes connaissances juridiques en matière de sécurité et de droit informatique
- Connaissance des normes ISO (si l'entreprise dispose d'une certification) et/ou PCI/DSF (banques, grande distribution ou e-commerce)
- Maîtrise de l'anglais, car 90 % des documents relatifs à la sécurité sont rédigés en anglais

Aptitudes professionnelles

- Sens de la confidentialité, intégrité et éthique car le responsable sécurité a accès à des informations sensibles et stratégiques pour l'entreprise
- Rigueur, capacité d'anticipation et sens de la méthode afin de mettre en place des programmes de sécurité efficients
- Pédagogie pour expliquer aux utilisateurs les règles à respecter pour ne pas mettre en danger le système d'information de l'entreprise
- Diplomatie, écoute, sens du dialogue, persuasion, pour convaincre les utilisateurs des risques encourus et du bien-fondé des procédures mises en place
- Résistance au stress pour faire face à des situations de crise nombreuses et inattendues (intrusion, virus, problème de sécurité « matérielle » (incendies, fuites d'eau…)) et à prioriser les actions à mener
- Curiosité, car le responsable sécurité doit, en permanence, se tenir au courant des nouveaux risques et des nouvelles parades (virus et antidotes)
- Force de proposition pour faire évoluer la stratégie, ainsi que les pratiques
- Capacité à travailler et à s'adapter à tous les niveaux d'interlocuteurs de l'entreprise en adaptant son langage et son niveau d'explication à la population avec laquelle il est amené à travailler

N°25 : Webmaster

Autres intitulés :

- Webmestre
- Webmaster éditorial
- Administrateur de site web

Définition :

Le webmaster a pour mission de gérer le site Internet et/ou intranet d'une entreprise en prenant en charge l'ensemble des aspects techniques et éditoriaux.

Activités principales :

Gestion technique du site

- Assurer les relations avec l'hébergeur du site.
- Concevoir les pages web ou participer à la rédaction du cahier des charges pour faire réaliser le site par un sous-traitant.
- Gérer les incidents techniques de premier niveau dans l'exploitation du site web.
- Participer au choix des sous-traitants éventuels.

Développement et/ou supervision des applications

- Rédiger les lignes de codes nécessaires à la création d'un produit web, pouvant contenir du texte, des images, de la vidéo et/ou du son.
- Appliquer les règles de navigation, créer des liens entre les pages.
- Corriger et optimiser les fonctionnalités (design du site, charte graphique, ergonomie...).
- Assurer les tests et recettes des applicatifs (développés par lui ou par des prestataires).
- Suivre les prestataires.
- Assurer ou superviser la maintenance du site.

Gestion de contenu éditorial et animation du site

- Réaliser l'intégration technique et graphique des contenus éditoriaux.
- Rédaction de newsletters.
- Gestion de la FAQ (foire aux questions).
- Animation de communautés (pages Facebook, compte Twitter...).

Profil :

Diplômes requis

- Diplômes universitaires Bac + 2 à Bac + 5 en informatique ou avec une spécialisation de webmaster
- Formation professionnelle niveau 3 ou certifications professionnelles au métier de webmaster de type AFPA, CNAM…

Durée d'expérience

Ce poste est accessible aux jeunes diplômés, en particulier à ceux qui ont une formation Bac + 4 ou 5 ; cependant, le plus souvent une expérience de 1 à 5 ans dans le développement de sites web est requise.

Certaines entreprises peuvent demander des cadres confirmés dont l'expérience compensera le niveau de formation initiale.

Compétences requises :

Compétences techniques

- Maîtrise des architectures web et des problématiques d'accessibilité
- Notions de sécurisation des contenus
- Bonne maîtrise de langages et d'outils de développement (.NET, J2EE, SQL, JSP, XSLT…)
- Maîtrise des outils bureautiques et d'édition web (HTML, JavaScript, Flash, dreamweaver, Visual Basic, PHP, Photoshop....)
- Connaissance des réseaux sociaux, des techniques d'écriture sur le Web et du *community management*
- Connaissance des règles éditoriales des sites web
- Notions des techniques du *webmarketing* (référencement, e-mailing...), d'acquisition et d'outils de mesure d'audience

Aptitudes professionnelles

- Polyvalence, créativité pour identifier les solutions techniques appropriées et gérer à la fois les problématiques techniques et éditoriales...
- Forte motivation pour l'informatique et pour l'apprentissage de nouveaux langages et systèmes
- Adaptabilité et force de proposition pour répondre aux demandes des différents interlocuteurs tout en assurant la cohérence du site
- Capacité d'adaptation et bonne expression orale car le webmaster est en contact avec des interlocuteurs variés au sein de l'entreprise : direction générale, direction du marketing, direction commerciale, direction de la communication, service informatique, prestataires (hébergeurs, agences web...)
- Créativité et réactivité, car le webmaster doit savoir réagir vite en fonction des besoins. Il est fréquemment seul dans l'organisation à exercer cette fonction (notamment en PME)
- Rigueur et organisation car il occupe une fonction où il est amené à être polyvalent et il doit éviter de se disperser pour rester efficace

Conseil en SI/Maîtrise d'ouvrage

N°26 : Consultant fonctionnel

Autres intitulés :

- Consultant assistance à maîtrise d'ouvrage
- Analyste fonctionnel
- Consultant en organisation
- Consultant SI
- Architecte fonctionnel / urbaniste fonctionnel

Définition :

Véritable interface entre la technique et le métier, le consultant fonctionnel accompagne son client dans son processus de modernisation et de développement technologique.

Activités principales :

Phase préparatoire (audit fonctionnel et organisationnel)

- Déterminer les interlocuteurs et les acteurs clés du projet.
- Auditer les processus concernés (interviews complémentaires, cartographie des processus, analyse des flux...).
- Étudier les solutions logicielles et/ou applicatives existantes.
- Mesurer les contraintes organisationnelles et technologiques de la société.
- Définir les impacts et les risques sur le fonctionnement global de l'entreprise.
- Évaluer les opportunités de faisabilité opérationnelle du projet.
- Préconiser les améliorations en termes de processus et/ou de solutions progicielles.

Participation à la rédaction du cahier des charges et à la sélection des prestataires

- Fixer le périmètre et les objectifs du projet (valeur ajoutée, planning, enveloppe budgétaire...).
- Recueillir et analyser les besoins fonctionnels auprès des départements « métier » concernés.
- Rédaction des spécifications fonctionnelles et du cahier des charges.
- Consultation des prestataires ayant répondu aux appels d'offre.
- Comparer et analyser les différentes solutions du marché.
- Sélectionner le ou les prestataire(s) et la solution logicielle.
- Participer à la formalisation du contrat et des plans d'assurance qualité.

Support à la mise en œuvre du projet

- Mettre en place les structures du projet et les règles de fonctionnement (plans de formation, définition des méthodologies et des outils de pilotage...).
- Réaliser les arbitrages et piloter l'avancement des travaux (choix des indicateurs, définition des étapes intermédiaires des projets...).
- Allouer et suivre l'affectation des ressources (répartition des charges, définition des objectifs individuels et/ou collectifs...).

- Superviser la mise en œuvre, le paramétrage et l'interfaçage du progiciel.
- Communiquer sur l'état d'avancement du projet auprès de la maîtrise d'ouvrage et des utilisateurs.

Test et recette fonctionnelle

- Définir les plans de test, les scenarii et les jeux d'essai.
- Effectuer les tests pour chacune des solutions applicatives et/ou logicielles.
- Identifier les erreurs et dysfonctionnements.
- Établir les plans de correction et d'amélioration avant la mise en production.
- Rédiger le rapport d'anomalies.

Validation et mise en production

- Vérifier la conformité de l'ouvrage par rapport à la demande formulée par la maîtrise d'ouvrage durant les phases de spécifications.
- Rédiger les référentiels et les documentations orientés utilisateurs.

N.B. : Les étapes de validation peuvent être nombreuses suivant les projets. Cette étape est particulièrement stratégique, car elle va permettre de définir de manière contractuelle les délais et les conditions de garantie.

Conduite du changement et formation des utilisateurs

L'intégration d'un nouvel outil dans l'entreprise implique généralement des changements dans le mode de fonctionnement et les habitudes de travail des salariés. Cette étape implique de prendre en compte la dimension humaine, au travers de la mise en place d'une démarche de participation, de communication et de formation et/ou coaching, autour des nouveaux modes de fonctionnement.

Profil :

Diplômes requis

- École supérieure de commerce ou diplôme Bac +4/5 en gestion
- École d'ingénieurs (informatique, télécoms, généraliste)
- Mastères spécialisés (grandes écoles d'ingénieur ou grandes écoles de commerce)

N.B. : Il existe aujourd'hui un grand nombre de mastères spécialisés permettant aux consultants d'acquérir une vision technico-fonctionnelle des systèmes d'information. Cette double compétence est particulièrement appréciée dans le monde du service (SSII, cabinets de conseil...) et dans les grands groupes internationaux.

Durée d'expérience

Au moins trois ans d'expérience professionnelle, soit en tant que consultant, soit dans un métier qui lui a permis d'acquérir une véritable expertise métier.

Compétences requises :

Compétences techniques

- Bonne culture technologique couplée à une vision globale des solutions applicatives et/ou logicielles
- Connaissance des fonctionnalités et des spécificités du métier et du système d'information de l'entreprise
- Maîtrise de l'ingénierie des processus
- Très bonne connaissance sectorielle (banque, medias, administration…) et/ou fonctionnelle (finances, relation clients, logistique…)
- Techniques de modélisation (UML, Merise) et méthodes de gestion de projet
- La maîtrise de l'anglais est indispensable sur des missions internationales

Aptitudes professionnelles

- Autonomie et curiosité d'esprit afin d'être force de propositions et anticiper les dérives fonctionnelles du projet
- Bon relationnel afin de savoir se fondre au sein des équipes de son client
- Qualités d'écoute et sens du client afin de comprendre et intégrer l'ensemble des besoins des utilisateurs et éviter l'élaboration d'une solution inadaptée
- Hauteur de vue afin de déterminer les besoins et les objectifs prioritaires
- Adaptabilité afin d'être en mesure d'intégrer rapidement les contraintes opérationnelles de son client
- Esprit d'analyse et de synthèse afin d'aller à l'essentiel afin de cadrer au mieux le périmètre de son projet
- Qualités pédagogiques pour accompagner et rassurer les utilisateurs tout au long du projet. Cette qualité s'illustre particulièrement lors des phases de conduite du changement.

N°27 : Consultant informatique décisionnelle/ BIG DATA

Autres intitulés :

- Chief data officer
- Responsable BI/big data
- Consultant BI (Business Intelligence)
- Responsable infocentre
- Consultant BI et Datawarehouse
- Spécialiste ETL (Extract Transform Loading)
- Data Scientist
- Expert BI/big data
- Analyste R&D big data

Définition :

L'expert en informatique décisionnelle collecte les données et les transforme en informations et outils d'aide à la décision. Il analyse des masses importantes de données éventuellement non structurées (big data), les visualisent et propose de nouveaux services aux utilisateurs.

Activités principales :

Définition d'un projet de S.I. décisionnel et/ou big data

- Concevoir une solution d'environnement d'aide à la décision après avoir mis en place de réunions/workshops avec les différents acteurs (systèmes sources, équipes de production, utilisateurs) afin de définir clairement les besoins/contraintes de chacun.
- Planifie et l'estime le coût du projet dans son ensemble.
- Coordonner les différents acteurs (utilisateurs, responsable métiers, équipes informatiques…) afin de définir le cadre du projet.
- Définir les choix techniques en termes de produit en fonction du S.I. existant.

Conception de l'architecture de l'entrepôt de données

- Concevoir l'architecture d'un entrepôt de données décisionnel (datawarehouse).
- Définir les solutions de stockage et la structuration des données au sein d'un modèle.
- Déterminer les outils d'acquisition de données depuis un ensemble de bases fonctionnellement et techniquement hétérogènes.
- Déployer des outils d'extraction de données en recherchant la pérennité, la fiabilité et l'évolutivité de ces outils.
- Étudier et mettre en place les meilleures solutions techniques pour gérer les gros volumes de données.
- Rédaction de règles (guidelines) pour la bonne mise en œuvre des technologies ETL (Extract Transform Loading)
- Réaliser les tests et recette techniques pour vérifier l'alimentation et la cohérence des données.

Configuration des outils d'analyse et de reporting

- Réaliser le recueil et la définition des besoins utilisateurs
- Rédiger les cahiers des charges formalisant les besoins des métiers et les spécifications fonctionnelles.
- Organiser les réunions de validation et hiérarchiser les besoins (besoins communs, spécifiques, degré d'urgence…).
- Définir les règles d'utilisation des technologies décisionnelles (use cases, arbres de décision...).
- Exploiter et valoriser des données en utilisant des techniques statistiques ou des algorithmes (big data).
- Concevoir les indicateurs et les calculer (en construisant des tables de type datamart).
- Intégrer les nouvelles données dans le reporting existant.
- Réaliser la recette technique et fonctionnelle de ces outils.

Restitution des données et formation des utilisateurs

- Développer les « univers » et les rapports.
- Définir des outils de *reporting* dynamique (OLAP, bases multidimensionnelles).
- Assurer la présentation des données selon les besoins de l'utilisateur.
- Assurer la formation des utilisateurs à l'utilisation des outils décisionnels.

Profil :

Diplômes requis

- Formation universitaire supérieure Bac + 5 (master 2 en management de projets, informatique, télécoms, statistiques, mathématiques…)
- Ecole d'ingénieurs (informatique, télécoms, généraliste…)

Durée d'expérience

Le poste de consultant B.I. est proposé prioritairement aux jeunes diplômés et aux jeunes cadres ; les postes de chef de projet, directeur de projets ou responsable B.I. sont accessibles aux cadres confirmés disposant au minimum de 5 ans d'expérience. En SSII, de jeunes cadres peuvent parfois se voir confier des fonctions de consultant manager. Les postes de data scientist ont tendance à se développer et sont confiés à des cadres ayant déjà une expérience professionnelle et notamment à ceux qui ont travaillé sur ces thématiques dans des laboratoires de recherche (universitaires ou centres R&D d'opérateurs télécoms).

Compétences requises :

Compétences techniques

- Connaissances générales en informatique : architecture des systèmes, bases de données, méthodologies de développement, CRM, ERP…
- Compréhension de l'environnement et des activités de l'entreprise, des besoins et des contraintes des utilisateurs
- Bonne connaissance du domaine fonctionnel dans lequel il intervient
- Maîtrise de l'organisation et de la conduite de réunions

- Maîtrise de l'expression de besoins
- Bonne connaissance de la méthodologie et des outils de tests
- Connaissance éventuelle d'ERP (Enterprise Resource Planning ou progiciels intégrés), de CRM (Customer Relationship Management ou gestion de la relation client)
- La maîtrise de l'anglais peut être nécessaire pour certains postes.

Spécifiques B.I. :

- Maîtrise du langage de requête SQL
- Maîtrise des outils d'alimentation (ETL) de type Informatica, Datastage…
- Maîtrise des outils de reporting statique de type Business Object (BO), Cognos… et de reporting dynamique (OLAP) comme Essbase ou MS OLAP

Spécifiques big data :

- Maîtrise des technologies Hadoop
- Excellente connaissance des bases de données No-SQL
- Bonnes connaissances en statistiques et en algorithmie
- Connaissance des usages marketing des données issues du web 2.0

Aptitudes professionnelles

- Qualités relationnelles, car le consultant informatique décisionnelle est en relation avec les spécialistes métiers, la direction générale de l'entreprise, la maîtrise d'ouvrage, les équipes de développement, la production informatique, la sécurité (RSSI) et les éditeurs de solutions
- Qualités rédactionnelles, pour rédiger les cahiers des charges
- Esprit de synthèse de manière à avoir une vision globale des résultats qui doivent être fournis
- Analyse, car il doit donner à ses interlocuteurs des éléments permettant de faire des choix en fonction de leurs urgences et du retour sur investissement attendu
- Capacité à vulgariser des sujets techniques complexes afin de permettre aux interlocuteurs des métiers de prendre en compte les problématiques techniques de l'informatique
- Diplomatie pour concilier des intérêts parfois divergents : ceux des métiers et ceux techniques et financiers

Spécifiques big data :

- Goût pour les nouvelles technologies, car le consultant travaille sur des technologies et des concepts récents
- Capacité à innover et à chercher des axes d'analyse inédits
- Force de conviction pour que les analyses soient utilisées par l'entreprise (big data)

N°28 : Consultant intégrateur de progiciel

Autres intitulés :

- Consultant ERP
- Consultant CRM
- Consultant SAP
- Consultant progiciel
- Chef DE projet ERP/CRM
- Directeur de projet intégration de progiciel

Définition :

Le consultant intégrateur de progiciel sert d'interface entre les équipes métier et l'éditeur du progiciel : il aide le client à spécifier ses besoins en fonction des possibilités du progiciel, paramètre l'outil en fonction des spécifications fonctionnelles, forme les utilisateurs.

Activités principales :

Analyse des besoins des utilisateurs

- Recueillir et analyser les besoins des équipes métiers pour de nouveaux projets ou des refontes importantes.
- Evaluer la partie du projet ne répondant pas au modèle standard et proposer des solutions.
- Analyser les impacts organisationnels.
- Chiffrer le coût des développements spécifiques.
- Rédiger les cahiers des charges formalisant les besoins des métiers ainsi que les spécifications fonctionnelles détaillées.
- Négocier avec les utilisateurs de manière à adapter leur besoins et les process aux standards du progiciel.

Participation à la mise en œuvre du projet

- Analyser la faisabilité technique des demandes métiers, ainsi que la cohérence avec l'urbanisme et l'architecture globale du S.I. avec les différents experts.
- Animer des réunions avec les équipes métiers de manière à affiner la demande.
- Identifier des incidences sur les autres applicatifs.

Participer au paramétrage du progiciel

- Suivre l'avancement du projet et répondre aux problèmes rencontrés par les développeurs en charge des développements spécifiques (ex. : précisions ou incohérences dans certaines spécifications, ou conflits avec d'autres modules).
- Gérer les problématiques liées à l'interaction avec la mise en œuvre des autres modules.
- Assurer le planning du projet.
- Piloter la recette fonctionnelle (tests utilisateurs).
- Coordonner le déploiement du projet (suivi de la mise en ligne et remontée des incidents ou dysfonctionnements).
- Assurer le suivi des corrections des incidents selon leur nature.

Mise en place de la conduite du changement

- Suivre le déploiement des nouveaux applicatifs auprès des utilisateurs, coordonner les relations avec les équipes de production informatique et assurer le support fonctionnel utilisateurs.
- Réaliser les actions de transfert des compétences aux équipes opérationnelles, préconiser et/ou assurer la formation des formateurs.
- Assurer le reporting du projet auprès du directeur du projet et des directeurs des entités concernées par le projet.
- Rédiger ou participer à la documentation destinée aux utilisateurs.

Profil :

Diplômes requis

- Formation de niveau Bac + 5 (master en gestion, organisation, management de projets, informatique, audit…)
- Ecole d'ingénieurs (informatique, télécoms, généraliste…)
- École supérieure de commerce
- IEP

Une certification dispensée par l'éditeur (SAP, Oracle, IBM…) peut être exigée.

Durée d'expérience

Le poste de consultant progiciel est proposé prioritairement aux jeunes cadres et éventuellement à des jeunes diplômés; les postes de chef de projet, consultant manager, directeur de projet intégration sont accessibles aux cadres confirmés disposant au minimum de 10 ans d'expérience.

Compétences requises :

Compétences techniques

- Connaissances générales en informatique : architecture des systèmes, bases de données, méthodologies de développement…
- Compréhension de l'environnement et des activités de l'entreprise, des besoins et des contraintes des utilisateurs
- Très bonne connaissance du domaine fonctionnel dans lequel il intervient
- Maîtrise des principes de fonctionnement des ERP (Enterprise Resource Planning ou progiciels intégrés), et des CRM (Customer Relationship Management ou gestion de la relation client).
- Excellente connaissance du produit et/ou sur lequel il intervient
- Connaissance éventuelle des autres modules ou d'autres progiciels
- Maîtrise de l'organisation et de la conduite de réunions
- Maîtrise de l'expression de besoins
- Bonne connaissance de la méthodologie et des outils de tests fonctionnels
- Maîtrise des outils bureautiques et de PMO (Project management office)
- Connaissance d'outils de BI (Business Intelligence) pour certains postes
- Maîtrise de l'anglais

Aptitudes professionnelles

- Qualités relationnelles, car le consultant progiciel est en relation avec les spécialistes métiers, l'éditeur, les équipes d'infrastructure et de sécurité (RSSI) et il assure l'interface entre les équipes de développement et les métiers
- Esprit de synthèse de manière à avoir une vision globale d'un projet
- Analyse, car il doit donner à ses interlocuteurs des éléments permettant de faire des choix en fonction de leurs urgences et du retour sur investissement attendu
- Intérêt pour les nouvelles technologies et les problématiques techniques
- Capacité à vulgariser des sujets techniques complexes afin de permettre aux interlocuteurs des métiers de prendre en compte les problématiques techniques du progiciel
- Capacité à travailler au sein d'une équipe pluridisciplinaire, car les progiciels touchent des domaines fonctionnels variés et interdépendants
- Diplomatie pour concilier des intérêts parfois divergents : ceux des métiers et ceux techniques et financiers

N°29 : Consultant technique

Autres intitulés :

- Expert technique
- Consultant métier (par exemple : consultant CRM / BI / E-Business / Internet / Sécurité / Infrastructure...)
- Urbaniste technique

Définition :

Le consultant technique est chargé de définir et de concevoir des solutions techniques pour le compte de clients utilisateurs : il contribue à la phase de construction de l'offre en amont (avant-vente) et en aval (réalisation du projet l'implémentation).

Activités principales :

Contribution à la construction de l'offre

- Assurer une veille technologique importante dans son domaine de spécialité.
- Se former aux nouvelles technologies (y compris par le biais de l'autoformation).
- Identifier les partenaires technologiques (éditeurs de logiciels, sociétés de conseil en système d'information, SSII...) compétents dans son domaine d'expertise technique et susceptibles d'apporter une contribution complémentaire à celle de son entreprise.
- Participer en relation avec le directeur de projet et le directeur du département Conseil à l'élaboration et au positionnement de l'offre.

Participation aux phases d'avant-vente

- Intervenir à la demande des ingénieurs commerciaux, des ingénieurs d'affaires et ingénieurs avant-vente lors des phases préalables à la signature des contrats.
- Apporter au client une expertise technique pointue.
- Participer à la rédaction de la réponse à l'appel d'offre.

Réalisation d'audits techniques et rédaction de préconisations

- Réaliser, lors des phases amont des projets, des audits techniques permettant d'analyser l'existant et de faire émerger des solutions (il pourra s'agir par exemple d'audit de sécurité afin de vérifier la solidité de la solution logicielle).
- Conseiller le client (maîtrise d'ouvrage ou architecte) dans les choix de ses infrastructures logicielles et matérielles.

Participation au déroulement du projet

- Apporter un appui technique aux équipes de développement, prendre en charge certains développements spécifiques.
- Participer à l'installation lorsqu'il s'agit de matériels ou de logiciels complexes.
- Garantir la performance des nouvelles applications.
- Jouer le rôle d'interface entre les équipes en charge du développement ou de l'intégration et les utilisateurs finaux.
- Prendre en charge à titre personnel certains développements.

Profil :

Diplômes requis

- École d'ingénieurs (informatique, télécoms, généraliste)
- DESS/DEA informatique
- Diplôme de type Bac +4 en informatique : MIAGE, IUP informatique, maîtrise informatique, ingénieur maître...

Durée d'expérience

Le poste de consultant technique s'adresse à des candidats disposant au minimum d'une expérience de trois ans.

Compétences requises :

Compétences techniques

- Très bonne compréhension du système d'information de l'entreprise (réseau, système, infrastructures, exploitation…)
- Bonnes connaissances des problématiques fonctionnelles (spécifications fonctionnelles) exprimées par les utilisateurs
- Excellente expertise technique dans le domaine technique concerné par le poste : très bonne connaissance des offres logicielles du marché, des systèmes d'exploitation associés, des langages de développement soutenant la conception des offres logicielles et des problématiques d'intégration de service
- Ouverture vers les technologies objet (type J2EE, ASP.net) qui sous-tendent aujourd'hui l'essentiel de l'offre logicielle
- La maîtrise de l'anglais est nécessaire pour comprendre les documentations techniques, une partie importante de l'offre logicielle venant des États-Unis. De nombreux séminaires, colloques, présentations se déroulent exclusivement en anglais.

Aptitudes professionnelles

- Ouverture d'esprit afin de prendre en compte et d'intégrer les évolutions relatives à son domaine d'intervention
- Rigueur et ténacité : le consultant technique ne doit pas s'arrêter à une vision superficielle du sujet. Il doit poursuivre sa réflexion pour pouvoir fournir une prestation fiable et de qualité. Son statut d'expert ne lui permet pas d'être approximatif sur son domaine d'intervention.
- Curiosité et ouverture d'esprit face aux évolutions récentes afin de préserver sa capacité à innover et de rester à la fois compétitif et motivé
- Bonnes aptitudes relationnelles avec le client, afin de le rassurer sur la crédibilité technique de l'offre et contribuer ainsi à l'obtention et au développement de l'affaire
- Qualités d'écoute aussi bien vis-à-vis du client que des équipes internes (développeur, consultant fonctionnel, ingénieur avant-vente…); capacité à travailler en mode projet
- Capacités pédagogiques pour expliquer l'offre en interne et en externe

N°30 : Responsable système d'information métier

Autres intitulés :

Responsable de domaine

Définition :

Le responsable d'un système d'information (SI) métier veille à la performance et à l'optimisation du SI pour un métier spécifique (logistique, commercial, relation client...) dans le respect des orientations stratégiques de l'entreprise.

Activités principales :

Analyse des besoins métier

- Dialoguer avec la direction de l'entreprise pour comprendre les grands axes de développement stratégique.
- Communiquer avec les utilisateurs métier pour comprendre leurs attentes fonctionnelles et leur degré de satisfaction par rapport au SI métier actuel.
- Faire le diagnostic des possibilités offertes par le système existant sur le plan quantitatif et qualitatif.
- Assurer une veille technologique afin de détecter les nouvelles possibilités techniques et fonctionnelles offertes par le marché.
- Participer à la description des besoins fonctionnels en amont de la rédaction du cahier des charges.
- Gérer quotidiennement le bon déroulement des projets relatifs au SI métier.
- Garantir le respect du cahier des charges, notamment en termes de qualité, budget, délais.

Définition des axes de développement en matière de SI métier

- Sélectionner les priorités de développement pour le SI métier en tenant compte de l'ensemble des contraintes économiques, techniques et humaines de l'entreprise.
- Superviser les phases de rédaction des spécifications fonctionnelles et des cahiers des charges.
- Fixer les objectifs de délais, budget et de performance qualitative et quantitative pour les projets mis en place.
- Allouer l'ensemble des ressources internes et externes nécessaires à la bonne réalisation des projets.
- Sélectionner et gérer les relations contractuelles avec les prestataires.

Pilotage de projet

- Superviser et coordonner le travail de l'ensemble des acteurs du système d'information (chefs de projet, ingénieurs développement, consultants...) et animer les équipes internes et/ou externes.
- Gérer les moyens humains nécessaires à la mise en place et au bon déroulement des projets, les sensibiliser aux besoins des utilisateurs finaux.
- Assurer l'interface avec les directions fonctionnelles utilisatrices du SI métier.

Accompagnement du changement et mesure de la performance

- Définir l'ensemble des moyens de communication interne nécessaires à la mise en place des nouveaux projets SI métiers : formation pour les utilisateurs, documents supports disponibles pour les utilisateurs...
- Mettre en place et suivre les tableaux de bord des performances du SI métier.
- Alerter l'équipe en charge du SI en cas de fonctionnement dégradé.
- Veiller à la cohérence de son SI métier avec l'ensemble des autres SI métier et celui de l'entreprise de manière générale.
- Anticiper les besoins futurs des utilisateurs, les évolutions du métier et prévoir les adaptations du SI métier en adéquation avec ces nouveaux besoins.

Profil :

Diplômes requis

- Écoles d'ingénieurs généralistes, spécialisées en informatique ou spécialisées dans le domaine d'activité de l'entreprise
- DESS / DEA informatique
- Écoles supérieures de commerce
- Formation spécialisée sur le métier de l'entreprise : 3es cycles ou cursus de type Bac +5 spécialisé

Durée d'expérience

Le poste de responsable du SI métier correspond à un niveau d'expérience de six à huit ans minimum.

Compétences requises :

Compétences techniques

- Bonnes connaissances générales informatiques : systèmes d'exploitation, outils de développement, principaux langages informatiques…
- Capacité à concevoir et à faire évoluer l'architecture d'un système d'information (conceptualisation et modélisation)
- Compréhension de l'environnement et des activités économiques de l'entreprise : concurrence, évolutions structurelles et conjoncturelles…
- Excellente connaissance des métiers de l'entreprise, des besoins et des contraintes des utilisateurs
- Maîtrise des techniques de gestion de projet (planning, budget, contrôle de gestion)
- Capacité à rédiger un cahier des charges et éventuellement des supports de formation à l'attention d'informaticiens et de non-informaticiens
- Maîtrise de l'anglais, sauf dans de rares cas où le responsable du SI métier évolue dans un environnement uniquement francophone

Aptitudes professionnelles

- Bonnes qualités relationnelles et de communication afin d'assurer une collaboration efficace avec son client interne ou externe
- Autonomie et confiance en soi pour gérer tous les aspects d'un projet
- Qualités d'animateur : écoute, dialogue pour animer et coordonner le travail de son équipe, et comprendre les besoins des utilisateurs finaux
- Excellente résistance au stress, car le responsable de SI métier est non seulement garant de la satisfaction des utilisateurs, mais également des performances économiques liées aux projets qu'il pilote
- Hauteur de vue pour réussir à dissocier les priorités à court terme des axes de développement stratégique à plus long terme
- Qualité de visionnaire, afin d'anticiper au mieux les évolutions du métier à long terme
- Force de conviction, car le responsable de SI métier doit souvent expliquer ses choix en matière de développement de nouvelles applications.

N°31: Urbaniste architecte fonctionnel système d'information

Autres intitulés :

- Architecte fonctionnel
- Architecte d'entreprise
- Enterprise architect

Définition :

L'urbaniste ou architecte fonctionnel garantit l'évolution et la cohérence de l'ensemble du système d'information d'une entreprise dans le respect de ses objectifs et de ses contraintes externes et internes. Il définit les règles d'urbanisation et veille à leur application. Il travaille en relation étroite avec l'architecte technique.

NB : L'architecte applicatif n'est pas traité dans cette fiche métier

Activités principales :

Conception, mise à jour et évolution du système d'information

- Construire, mettre à jour et faire évoluer la cartographie du système d'information de l'entreprise.
- Faire l'inventaire des fonctions informatiques existantes (applications et référentiels de données) et des flux d'informations.
- Identifier et modéliser les activités, les processus et les services à rendre.
- Définir un plan d'urbanisation cible.
- Proposer des scénarios d'évolution et de simplification du système d'information.

Garantie de la cohérence du S.I.

- Evaluer la pertinence et la cohérence des projets par rapport à l'architecture cible et aux systèmes existants.
- Mettre en évidence si nécessaire la complexité du S.I et les systèmes non-rationnels (flux multiples, redondances de données).
- Définir des cibles d'architecture fonctionnelle, applicative et technique.
- S'assurer que tous les projets d'évolution informatique sont alignés avec l'architecture cible.

Gestion des projets d'architecture

- Participer à l'amélioration ou à la création de nouvelles applications ou services en appliquant les principes d'urbanisation retenus.
- S'assurer de l'alignement entre les besoins et les solutions proposées.
- Mettre à jour les référentiels documentaires.

Conseil et aide à la décision

- Informer les différents acteurs du système d'information (direction informatique, MOA et directions métiers) et la direction générale des évolutions technologiques proposées.
- Participer aux études de décommissionnement d'applications.

Participation aux règles de gouvernance du système d'information

- Déterminer pour chaque domaine fonctionnel les outils les plus adaptés et accompagner leur mise en place et leurs évolutions.
- Mettre en place un processus de gouvernance d'architecture afin de s'assurer que les solutions implémentées sont conformes au système d'information cible.
- Superviser les développements effectués par les fournisseurs, notamment sur les interfaces entre applications.
- Piloter et réaliser des missions transverses sur l'évolution des modèles informatiques tant sur le plan organisationnel que fonctionnel ou technique.

Veille technologique

- Analyser les développements technologiques les plus récents.
- Rechercher des solutions innovantes pour l'intégration de nouvelles applications ou la création de nouvelles solutions.

Profil :

Diplômes requis

- Formation de niveau Bac +5 (master) spécialisée en informatique et/ou télécoms, sécurité des systèmes informatiques…
- École d'ingénieurs (informatique, télécoms, généralistes...)

Durée d'expérience

Au moins 10 ans d'expérience sont généralement requis car il s'agit de postes nécessitant une certaine maturité ainsi qu'une excellente connaissance des systèmes d'information et des processus métiers.

Compétences requises :

Compétences techniques

- Bonne connaissance de la stratégie de l'entreprise, de son organisation, de ses métiers et des enjeux
- Bonne connaissance du système d'information global, de la gouvernance du S.I.
- Excellentes connaissances des principes d'urbanisation du S.I.
- Connaissance des méthodes de modélisation des organisations et des processus
- Pratique de la mise en œuvre de référentiels méthodologiques (ITIL, COBIT)
- Connaissance d'outils de modélisation (UML …)
- Connaissance des méthodologies de cartographie
- Connaissance de méthodologies de frameworks d'architecture
- Bonnes connaissances en matière de sécurité et de droit informatique
- Maîtrise des outils bureautiques et de PMO (Project management office)
- Connaissance éventuelle d'ERP (Enterprise Resource Planning ou progiciels intégrés), de CRM (Customer Relationship Management ou gestion de la relation client) ou d'outils de BI (Business Intelligence)

- La maîtrise de l'anglais peut être nécessaire pour certains postes, dans la mesure où la plupart des postes concernent des entreprises de taille importante, ayant fréquemment des activités à l'international

Aptitudes professionnelles

- Sens de la confidentialité et éthique car l'urbaniste S.I. a accès à des informations stratégiques pour l'entreprise
- Rigueur, capacité d'anticipation et sens de la méthode afin de mettre en place des programmes de sécurité efficients
- Qualités relationnelles et diplomatie, car l'urbaniste est en relation avec la direction informatique, les directeurs des différents métiers, la maîtrise d'ouvrage, la maîtrise d'œuvre et la sécurité (RSSI) et assure l'adéquation entre les aspects purement système d'information et les aspects organisationnels financiers et humains
- Qualités rédactionnelles pour rédiger les cartographies, les processus, les documents d'architecture
- Esprit de synthèse et capacité à prioriser les problématiques de manière à avoir une vision globale du système d'information de l'entreprise et de la cible à atteindre
- Analyse, car il doit pouvoir comprendre les différentes interactions entre les divers systèmes informatiques de l'entreprise

N°32 : DATA SCIENTIST – DATA MINER

Autres intitulés :

- Ingénieur *data scientist*
- Ingénieur *big data*
- *Analyst dataminer*
- Consultant *dataminer*
- *Data analyst*

Définition :

Au service du marketing, le dataminer ou datascientist valorise l'ensemble des données client pour en faire un levier de création de valeur pour l'entreprise. Il analyse des masses de données hétérogènes, éventuellement non structurées, pour en extraire de la connaissance utile à l'optimisation des offres et services de l'entreprise.

Activités principales :

Extraction, uniformisation et structuration des données clients

- Collecter, sélectionner et valider les données clients pertinentes pour l'analyse.
- Définir les solutions de stockage et la structuration des données.
- Convertir, coder et cartographier des données de consommation ou d'usage produit dans un format compréhensible par l'ensemble des collaborateurs.
- Améliorer la qualité et enrichir les bases de données clients de l'entreprise.
- Déterminer les outils et méthodes d'acquisition de données depuis un ensemble de bases techniquement hétérogènes.
- Concevoir l'architecture d'un entrepôt de données décisionnelles (*Data warehouse*).
- Maîtriser la qualité des données tout au long de leur traitement.

Analyses prédictives et développement de la connaissance client

- Mettre en œuvre et garantir la modélisation statistique des données.
- Développer des algorithmes d'apprentissage et scénarios prédictifs des comportements clients.
- Optimiser la segmentation client à l'aide des statistiques et données de consommation.
- Étudier et mettre en place les meilleures solutions techniques pour gérer les grands volumes de données.
- Concevoir des modèles de détection des *insights* (Attitude ou croyance profonde de consommateurs, qui joue comme un frein ou comme une motivation à un comportement, sur laquelle le marketing va chercher à agir pour définir une offre ou construire une promesse publicitaire) et des opportunités de marché.
- Tester, contrôler la qualité et la cohérence des bases de données.
- Accompagner l'entreprise dans le développement de leviers de création de valeur.

Optimisation des actions marketing de l'entreprise

- Mettre en œuvre et optimiser les stratégies CRM du marketing relationnel.
- Améliorer la performance des plateformes de contact client (Web, etc.).
- Optimiser le ciblage des campagnes de marketing direct (emailing, sms, web, etc.).
- Analyser les taux de rétention client et les éléments y contribuant.
- Mesurer le ROI (Return On Investment ou retour sur investissement) de l'ensemble des actions marketing de l'entreprise.
- Construire et optimiser les scores d'appétence (outil qui traduit la probabilité qu'un prospect devienne un consommateur du produit ou service promu).
- Fournir au service marketing les données nécessaires à la réalisation d'études de marché.

Développement d'outils de support aux clients internes

- Participer à la mise en œuvre de la stratégie marketing de l'entreprise.
- Analyser l'ensemble des données commerciales pour développer des systèmes efficaces d'aide à la décision.
- Participer au développement des indicateurs de performance commerciale de l'entreprise.
- Fournir aux chefs de produit des leviers statistiques décisionnels pour la conduite et l'analyse des campagnes de prospection.
- Réaliser des études statistiques pour les clients internes ou la direction générale.
- Animer les ateliers d'expression des besoins internes et rédiger les cahiers des charges.
- Écrire et rédiger la spécification des besoins à destination des DSI ou de la maîtrise d'ouvrage.
- Déterminer les outils de reporting dynamique et multidimensionnel (OLAP).
- Présenter les résultats des études réalisées aux clients internes.
- Former les utilisateurs aux outils informatiques et décisionnels.

Veille technologique sur les outils de datamining

- Effectuer une veille sur les nouvelles technologies et solutions logicielles d'analyse des données.
- Rechercher et expérimenter de nouvelles méthodes de modélisation et d'analyse des données.

- Sélectionner les nouveaux outils et techniques de *data management*.

Management d'équipe

- Animer les réunions, organiser et planifier les interventions d'une équipe.
- Assurer le recrutement et le développement des compétences des collaborateurs.
- Gérer un budget et évaluer le coût des interventions.
- Dimensionner les projets et définir les choix techniques et méthodologiques des interventions.

Profil :

Diplômes requis

- Diplôme universitaire Bac +5 (Master en statistiques et marketing, informatique, statistique et informatique décisionnelle, économétrie)
- Diplôme d'école d'ingénieurs
- Master spécialisé *Big data analyse*, management et valorisation responsable
- Doctorat en informatique, en mathématiques, en statistiques ou en modélisation des données

Durée d'expérience

- Pour un poste junior : entre 1 et 5 ans d'expérience
- Pour un poste senior : 5 à 10 ans d'expérience

Compétences requises :

Compétences techniques

- Excellente maitrise des algorithmes d'apprentissage automatique (Machine Learning)
- Excellente maitrise des outils de data management (SAS, SPSS, SAP Infinite Insight, Python, R, Excel, Access...)
- Excellente maîtrise des technologies HADOOP
- Excellente maîtrise des bases de données SQL et no-SQL
- Bonne connaissance des réseaux de neurones et d'intelligence artificielle
- Bonne connaissance des outils de *Web analyse* (Omniture, Google analytics etc.)
- Solides connaissances en marketing
- Maîtrise de l'anglais

Aptitudes professionnelles

- Esprit d'analyse indispensable pour identifier et comprendre les problématiques économiques transversales de l'entreprise
- Très grande rigueur et forte concentration pour assurer l'exactitude des calculs réalisés
- Bonne capacité d'organisation pour structurer ses méthodes de travail et son plan d'intervention
- Sens aigu du service pour optimiser et améliorer la satisfaction du client
- Excellente capacité de communication pour expliquer et convaincre
- Très bonne pédagogie pour expliciter sa méthode de travail aux collaborateurs
- Curiosité pour suivre les nouvelles tendances et découvrir de nouveaux outils

- Goût pour les données numériques et les chiffres car les volumes de données sont importants
- Force de proposition pour proposer des recommandations aux équipes marketing
- Sens des affaires pour détecter les opportunités de marché
- Qualité d'écoute afin de recueillir avec précision les informations et besoins des clients internes et externes

Commercial/ Marketing

N°33: Directeur commercial

Autres intitulés :

- Responsable commercial
- Directeur des ventes
- Directeur régional
- Directeur international

Définition :

Le directeur commercial informatique a pour mission de définir une stratégie commerciale puis d'animer, coordonner et contrôler la mise en œuvre de la politique commerciale.

Activités principales :

Définition de la stratégie commerciale

- Définir le positionnement de l'entreprise et de ses produits sur le marché à partir des études qualitatives et quantitatives et des retours des forces de vente ; faire valider cette définition par la direction générale.
- Analyser les pratiques de concurrents.
- Détecter les grands comptes et les marchés potentiels.
- Faire émerger les axes de développement commercial de l'entreprise : segments porteurs, canaux de commercialisation, en particulier répartition des objectifs et des moyens entre vente directe et indirecte, définition des terrains de prospection et des gisements de croissance, etc.
- Appréhender auprès des équipes marketing et technique les fonctionnalités et les usages de l'offre produits et services.
- Participer à un ajustement de l'offre (en particulier en matière de maintenance, support client...)
- Définir les grandes lignes de l'argumentaire de vente.
- Définir les objectifs de commercialisation (volumes et marges).
- Définir le budget de fonctionnement de la direction commerciale en liaison avec le directeur général et la direction administrative et financière.
- Définir les moyens de commercialisation et les objectifs par zone, par secteur et par équipe.

Mise en œuvre de la politique commerciale

- Diriger les équipes commerciales : recruter les ressources humaines nécessaires à l'accomplissement de la stratégie, fédérer une équipe autour d'un projet, motiver les équipes en place et prévoir les formations leur permettant de s'adapter aux évolutions produits / marchés ou aux nouvelles techniques de vente.
- Définir la politique de rémunération, en particulier la partie variable, en liaison avec le département Ressources humaines, régler les conflits au sein des équipes.
- Participer directement à certaines négociations auprès de clients stratégiques et de certains prescripteurs clés (SSII, sociétés de conseil...)

- Prendre en charge, le cas échéant personnellement, certains appels d'offres importants.
- Accompagner périodiquement les ingénieurs commerciaux sur le terrain afin d'appréhender les besoins des clients et de former les nouvelles recrues.

Suivi et analyse des performances commerciales

- Suivre et analyser les résultats commerciaux globaux, par équipe, par canal, par zone géographique et par collaborateur.
- Assurer un *reporting* auprès de la direction générale sur la base de tableaux de bord (chiffre d'affaires et rentabilité) et d'analyses qualitatives (positionnement).
- Réorienter si nécessaire le plan d'action commerciale.

Profil :

Diplômes requis

- BTS Action commerciale / DUT Force de vente
- École de commerce
- École d'ingénieurs (informatique, télécoms)
- Autres écoles d'ingénieur généraliste

N.B. : On constate une élévation du niveau de qualification initiale requis pour accéder à ce poste et un recul des formations supérieures courtes.

Durée d'expérience

Une expérience minimum de huit ans est requise pour accéder à ces fonctions.

Compétences requises :

Compétences techniques

- Expertise des techniques commerciales, notamment des méthodes de négociation ; bonne pratique personnelle en tant que négociateur
- Connaissance générale de l'industrie informatique, des systèmes d'information, des architectures et des environnements techniques (une culture informatique assez large est exigée afin d'acquérir une légitimité interne ou externe, ce qui explique la proportion assez importante d'ingénieurs à ces fonctions)
- Bonne connaissance des prescripteurs, des utilisateurs et des décideurs ainsi que des relais d'opinion (presse spécialisée, sites Internet à vocation éditoriale dans le domaine du high-tech, clubs informatiques…)
- Bonne connaissance des produits, services et technologies associés
- Maîtrise des techniques de management, de motivation des équipes et de formation à l'acte de vente
- Outils de prospection (achat ou location de fichiers, bases de données clients, logiciels de suivi des ventes…)
- Très bon niveau d'anglais, car les marchés sont de plus en plus internationaux

Aptitudes professionnelles

- Sens de l'écoute et du dialogue afin d'animer et de coordonner le travail de son équipe
- Dynamisme et enthousiasme afin de motiver ses collaborateurs et de faciliter la relation clientèle
- Empathie afin de susciter l'adhésion aussi bien en interne (équipe commerciale et autres départements de l'entreprise), qu'en externe (clients, prescripteurs, médias…)
- Ouverture d'esprit et curiosité afin d'appréhender les évolutions des marchés et des techniques
- Sens des responsabilités et rigueur pour conduire de manière efficace une politique commerciale
- Recul, hauteur de vue et esprit de décision pour prendre les bonnes orientations stratégiques
- Goût pour les chiffres et compétences en gestion, afin de mesurer le retour sur investissement commercial
- Résistance au stress, notamment en ce qui concerne l'atteinte des objectifs

N°34: Ingénieur d'affaires

Autres intitulés :

- Ingénieur technico-commercial
- Chargé d'affaires
- Responsable d'affaires

Définition :

L'ingénieur d'affaires informatique analyse les besoins du client, propose des solutions adaptées techniquement et vérifie la bonne exécution de la prestation ou du projet. Il gère et développe un portefeuille de clients.

Activités principales :

Prospection et conquête de nouveaux clients

- Assurer une veille économique sur le marché afin de détecter les opportunités commerciales.
- Suivre et anticiper les offres des concurrents afin d'identifier leurs avantages et faiblesses.
- Présenter l'entreprise, ses produits, ses services.
- Répondre aux appels d'offre en partenariat avec les équipes techniques de l'entreprise et/ou un ingénieur avant-vente.
- Analyser les besoins et les projets et proposer des solutions sur les plans technique, humain et financier.
- Rédiger les propositions commerciales et assurer l'ensemble des phases de négociation et de signature du contrat d'intervention.

Suivi des projets et gestion des équipes techniques

- Assurer tout au long du projet un soutien fonctionnel et technique aux équipes en charge de la réalisation.
- Coordonner les équipes de support à la vente.
- Affecter ou demander l'affectation des ingénieurs en fonction des projets des clients et des plannings de missions.
- Rechercher l'adéquation des exigences du client externe avec les souhaits des clients internes (ingénieurs en développement).
- Résoudre les éventuels conflits entre le client et les équipes de production (ingénieurs et/ou consultants).
- Suivre les missions des ingénieurs affectés et gérer leur formation et leur évolution professionnelle.
- Assurer le reporting auprès du client en lui présentant des indicateurs pertinents sur l'avancée du projet et/ou de la prestation.

Développement et gestion du portefeuille clients

- Entretenir les relations avec les clients en tant qu'interlocuteur privilégié.
- Communiquer auprès des clients sur les nouvelles offres de service de l'entreprise : évolution de la prestation, évolution des technologies traitées...
- Assurer une veille technologique afin d'optimiser la qualité de la prestation et d'anticiper les demandes des clients.

- Détecter les opportunités de nouveaux contrats en évaluant en permanence les besoins du client.
- Négocier les contrats et les éventuels renouvellements.

Profil :

Diplômes requis

- École d'ingénieurs (informatique, télécoms, généraliste)
- École supérieure de commerce
- Diplôme de type Bac +4 en informatique : MIAGE, IUP informatique, maîtrise informatique, ingénieur maître...
- Diplôme de type Bac +2 en informatique : BTS/DUT informatique
- DESS/DEA informatique

Il faut noter que les doubles formations, école d'ingénieurs associée à un 3e cycle en marketing/vente, sont particulièrement appréciées.

Durée d'expérience

De deux à dix ans en fonction du profil et de la taille des projets à superviser.

Compétences requises :

Compétences techniques

- Une double compétence (technique et commerciale)
- Des connaissances techniques sont essentielles afin de pouvoir comprendre la demande du client et analyser ses besoins
- Pour certains produits, des compétences fonctionnelles sont nécessaires telles que comptabilité, finance et gestion
- En fonction de la formation initiale, les recruteurs s'attacheront à vérifier la sensibilité technique des candidats de formation commerciale et les prédispositions commerciales des candidats de formation technique. Il peut s'agir de commerciaux qui évoluent vers l'informatique ou l'inverse.
- Connaissance générale de l'industrie informatique, des systèmes d'information, des architectures et des environnements techniques
- Bonne connaissance des services proposés à la vente et sens du service
- Maîtrise des techniques de négociation et des techniques de vente
- La maîtrise de l'anglais technique est le plus souvent nécessaire. L'ingénieur doit pouvoir maintenir la relation commerciale avec des clients étrangers. Une expérience internationale est un plus.

Aptitudes professionnelles

- Bonnes qualités relationnelles afin de développer des contacts privilégiés avec ses clients, en phase de prospection comme lors du développement du compte
- Qualités d'écoute afin de comprendre les besoins des clients et des prospects
- Dynamisme et réactivité afin de s'adapter à l'évolution constante des besoins des clients et de faire face à la pression de la concurrence
- Force de persuasion afin de défendre ses propositions techniques et la qualité de la prestation proposée ou effectuée
- Mobilité : cette fonction nécessite souvent de fréquents déplacements, en France et à l'étranger

N°35 : Ingénieur commercial

Autres intitulés :

- Responsable de compte (ou *account manager*)
- Responsable grand compte
- Ingénieur grand compte

Définition :

L'ingénieur commercial informatique doit constituer, gérer et développer un portefeuille de prospects et de clients sur une zone géographique, un segment de clientèle ou un secteur d'activité, afin de réaliser le chiffre d'affaires et les marges bénéficiaires.

Activités principales :

Définition des cibles commerciales

- Analyser le marché, la concurrence et le portefeuille confié.
- Définir et faire valider les objectifs de conquête de nouveaux clients selon les secteurs d'activité et les régions.
- Identifier les comptes clés au sein du portefeuille.
- Proposer et faire valider par sa hiérarchie les moyens de prospection et de développement du portefeuille.
- Identifier et faire valider les principaux " revendeurs " en fonction de la politique de vente indirecte définie par son entreprise.

Conquête de clients

- Actionner les moyens de prospection définis en amont : prospection téléphonique directe, animation d'une force de télé-vente interne et externe, représentation de son entreprise sur les salons professionnels et les autres événements de la profession, etc.
- Répondre aux appels d'offre, rédiger les propositions commerciales en mobilisant d'autres services de l'entreprise (projets, affaires, équipes d'avant-vente, département technique, service marketing, etc.).
- Présenter aux prospects l'offre de produits et de services et les atouts de l'entreprise sur son marché.
- Écouter, analyser et reformuler les besoins des clients dans le cadre d'une proposition.
- Négocier avec eux les conditions tarifaires.
- Renouveler l'éventuel réseau de vente indirecte en sélectionnant et motivant de nouveaux apporteurs d'affaires.
- Argumenter, négocier les conditions tarifaires et « clôturer » la vente.

Gestion du portefeuille clients

- Entretenir les contacts avec les principaux clients et leur présenter régulièrement les évolutions de l'offre.
- Assurer une relation chez le client aussi bien avec les prescripteurs (directions fonctionnelles, utilisateurs des solutions...) qu'avec les décideurs (directeurs des achats, directeurs informatiques, directeur général, etc.).
- Négocier les contrats et les éventuels renouvellements auprès des décideurs.

- Mettre en avant et vendre aux clients des services complémentaires (maintenance, formation, etc.).

Reporting auprès de la hiérarchie et des autres départements

- Assurer une remontée de l'information du terrain auprès des différents services de l'entreprise, en particulier auprès du marketing.
- Organiser auprès de sa hiérarchie un reporting régulier sur les moyens déployés et les résultats obtenus (rapport de visite).

Profil :

Diplômes requis

- Diplôme Bac +2 informatique ou commercial : DUT/BTS informatique, force de vente ou action commerciale
- École supérieure de commerce
- DESS et maîtrise marketing, gestion, économie
- École d'ingénieurs (informatique, télécoms, généraliste)

Durée d'expérience

Le poste d'ingénieur commercial s'adresse à un cadre ayant au moins deux ans d'expérience professionnelle.

Compétences requises :

Compétences techniques

- Connaissance générale de l'industrie informatique, des systèmes d'information, des architectures et des environnements techniques
- Connaissances techniques essentielles afin de comprendre la demande du client et d'analyser ses besoins
- Dans le cadre de la commercialisation de certaines offres, des compétences fonctionnelles sont nécessaires
- Bonne connaissance des produits et services commercialisés
- Maîtrise des techniques de négociation et des techniques de vente
- Bon carnet d'adresses, aussi bien dans le monde des entreprises que dans celui des associations et groupements professionnels (activités extraprofessionnelles, loisirs personnels, peuvent également être mises à profit pour élargir le réseau)
- Quelques bases en gestion
- Maîtrise de la micro-informatique, notamment des tableurs
- La maîtrise de l'anglais technique correspond au minimum requis. De plus en plus souvent, un anglais courant est demandé et une culture internationale est exigée.

Aptitudes professionnelles

- Bonnes aptitudes relationnelles afin de développer des contacts privilégiés avec les clients
- Écoute afin de comprendre les besoins des clients et des prospects
- Capacités à analyser et reformuler les besoins des clients afin de rédiger le cahier des charges et faire émerger des préconisations
- Bonne présentation car l'apparence demeure importante dès lors qu'il existe un contact avec les clients
- Dynamisme, pro-activité afin de s'adapter à l'évolution constante des besoins des clients et de l'environnement et de saisir les opportunités présentées par le marché
- Persévérance et résistance à la frustration car le commercial est souvent confronté à l'échec
- Sens du client et de la satisfaction client, car de celle-ci dépend le développement du chiffre d'affaires
- Capacité à argumenter et à convaincre le client, notamment dans la phase finale de la vente
- Mobilité géographique car cette fonction nécessite de fréquents déplacements en France et à l'étranger

N°36: Chef de produit technique

Autres intitulés :

- *Product manager*
- Ingénieur marketing produit
- Ingénieur produit
- Responsable ou spécialiste produit

Définition :

Le chef de produit technique en informatique gère la vie d'un produit, de sa conception à sa commercialisation, en lien avec les équipes techniques et commerciales de l'entreprise.

Activités principales :

Analyse du marché

- Surveiller les évolutions technologiques et leurs incidences sur son marché grâce notamment à des études qualitatives et quantitatives.
- Assurer une veille concurrentielle.
- Agréger l'ensemble des informations issues des services commerciaux, marketing et technique afin d'optimiser la connaissance du marché.
- Anticiper les besoins des entreprises clientes en termes de technologies et de fonctionnalités des produits.

Définition et conception de l'offre

- Définir une offre de produits ou de service adaptée à la demande du marché.
- Réaliser une étude de faisabilité du produit ou des services.
- Analyser et optimiser le produit ou service en fonction des contraintes propres à l'entreprise (ressources humaines, coûts, rentabilité, image de l'entreprise) et de la demande du marché.
- Mettre en place un business plan étudiant l'ensemble du positionnement du produit et sa rentabilité.
- Adapter en permanence l'offre à l'évolution et aux opportunités du marché.

Développement du produit et interface avec les équipes techniques

- Définir des spécifications fonctionnelles précises et assurer la bonne traduction en spécifications techniques avec les équipes de développement.
- Suivre le plan de développement du produit (" *roadmap produit* ") avec les équipes techniques.
- Valider l'adéquation des développements avec le cahier des charges défini en amont.
- Suivre l'avancée du développement en lien avec les impératifs de lancements commerciaux, le " *time to market* ".
- Assurer le suivi budgétaire et l'adéquation permanente des ressources aux objectifs fixés.

Pilotage et accompagnement du lancement commercial

- Élaborer une stratégie de commercialisation du produit ou du service : prix, promotion...
- Définir des services associés à l'offre principale : installation, maintenance, formation...
- Concevoir l'ensemble des supports techniques d'aide à la vente, soit à destination des équipes commerciales, soit à destination des équipes de communication en charge des actions de lancement du produit.
- Assurer des actions de communication en interne afin de présenter le produit : formation des équipes commerciales, présentation aux départements stratégiques de l'entreprise.
- Suivre l'évolution des ventes en permanence afin d'ajuster la stratégie marketing.

Profil :

Diplômes requis

- École d'ingénieurs (informatique, télécoms, généraliste)
- DESS / DEA informatique
- 3e cycle en marketing, type Mastère d'école de commerce, ou DESS marketing
- École supérieure de commerce

Les doubles formations sont très appréciées pour les postes de chef de produit technique : idéalement, une formation d'ingénieur doublée d'un 3e cycle en marketing.

Durée d'expérience

Une expérience de deux à cinq ans est généralement requise pour accéder à la fonction. Elle peut toutefois être accessible à de jeunes diplômés qui sont alors rattachés à un chef de produit senior.

Compétences requises :

Compétences techniques

- Maîtrise des différents éléments du mix marketing : prix, produit, promotion, distribution
- Maîtrise des techniques de gestion de projet : cycle en V, méthode agile...
- Connaissance des principaux langages et outils informatiques liés au développement du produit, afin de favoriser les échanges avec les équipes techniques
- Bonne culture générale du secteur d'activité de l'entreprise
- Compétences managériales, notamment dans le cadre d'une relation fonctionnelle avec la plupart des départements de l'entreprise
- Anglais courant afin de ne pas limiter le développement du produit à un marché francophone

Aptitudes professionnelles

- Force de persuasion afin de convaincre ses interlocuteurs de la qualité de l'offre proposée
- Bonnes qualités relationnelles afin de jouer un rôle d'interface avec les différents intervenants sur le projet
- Bonnes capacités d'analyse, car la compréhension d'un marché est essentielle dans le positionnement amont d'un produit
- Organisation et sens du délai afin de coordonner l'activité des différents services
- Qualités de communication, que ce soit dans le cadre de la formalisation des outils d'aide à la vente ou dans le cadre de présentations commerciales du produit
- Curiosité, notamment sur le plan technique, afin de favoriser les activités de veille et le dialogue avec les équipes de développement
- Pragmatisme, afin d'évaluer le positionnement du produit aussi bien sur des critères techniques que financiers, en tenant compte de la stratégie de l'entreprise

N°37: Chef produit web/mobile

Autres intitulés :

- *Internet product manager*
- Chef de produit digital web et mobile
- Chef de produit marketing web
- Chef de projet mobil

Définition :

Le chef de produit web/mobile a pour mission de définir, concevoir et mettre en œuvre de nouveaux services digitaux web et mobile ou d'améliorer l'offre produit existante dans le respect de la stratégie marketing de l'entreprise.

Activités principales :

Mise en place d'une veille concurrentielle et technologique

- Comprendre et évaluer les tendances actuelles, les nouveaux besoins des utilisateurs, et notamment ceux en lien avec les réseaux sociaux.
- Analyser les services offerts par les sites concurrents (benchmark).
- Mesurer l'impact des nouvelles technologies ou des nouveaux usages sur les produits et services (existants ou en cours de développement).

Conception et développement de nouveaux services web ou mobile

- Proposer de nouveaux services à partir de *brainstorming*, de recueil de besoins internes et d'études de marché, en prenant également en compte les contraintes techniques, financière et juridiques.
- Veiller à assurer la cohérence et l'interaction des services sur le site.
- Définir les cibles utilisateurs.
- Définir le cahier des charges et rédiger les spécifications fonctionnelles ou *users story*.
- Définir l'ergonomie des pages web présentant les services de manière à définir et à optimiser les parcours clients.
- Elaborer un plan produit et le suivi opérationnel de la *road map* (planning de lancement) en collaboration avec les services marketing et/ou communication.
- Suivre l'évolution des développements, faire les arbitrages et réaliser les tests utilisateurs
- Valider les applicatifs livrés avec la maîtrise d'ouvrage.

Lancement des nouveaux services web ou mobile

- Planifier et coordonner avec la direction commerciale et marketing les différentes phases de mise en œuvre des produits et/ou services.
- Rédiger la documentation destinée au marketing opérationnel et aux autres services (services commerciaux, service client, service juridique...).
- Coordonner la promotion, les programmes d'affiliation, les opérations spéciales avec les directions communication et commerciale.

Suivi des actions

- Suivre les statistiques grâce aux outils de webanalyse et de datamining.
- Analyser les réactions des internautes grâce à des sondages ou des enquêtes.
- Définir les KPI (*key performance indicators*), les suivre et les analyser pour s'assurer du fonctionnement des nouveaux produits, relever les dysfonctionnements.
- Faire le reporting (quantitatif et qualitatif) au directeur du marketing : nombre de pages lues, téléchargements, taux de transformation, nombre de fans...
- Assurer le suivi du budget.

Optimisation des services

- Concevoir les actions correctrices et définition de nouvelles spécifications.
- Suivre sur le long terme les interactions de l'internaute avec la marque ou le service et définir de nouvelles actions pour relancer la visibilité du service.

Profil :

Diplômes requis

- Formation supérieure de type IEP, école de commerce ou d'ingénieurs complétée par un master en webmarketing.
- Formation de niveau Bac +5 en marketing (master en webmarketing)
- Formation d'ingénieurs

Durée d'expérience

Le poste de chef de produit web/mobile est ouvert aux jeunes diplômés disposant d'une courte expérience sur des projets digitaux, web ou mobiles. Néanmoins, la dimension technique associée aux projets mobiles étant plus forte, une courte expérience dans le mobile peut être requise.

Compétences requises :

Compétences techniques

- Bonne connaissance de la sociologie des internautes et l'univers digital : usages (notamment en termes de navigation) et environnement socioculturel des membres (langage, codes « sociaux », jargon...)
- Bonne connaissance de ses concurrents
- Maîtrise des techniques de veille et de recherche permettant de fournir de l'information, de détecter les nouvelles tendances
- Bonne capacités rédactionnelles et rigueur dans la rédaction des spécifications
- Connaissance des techniques du webmarketing (stratégies de contenus, *search marketing*, affiliation, marketing viral, marketing mobile...), d'acquisition de trafic et d'outils de mesure d'audience
- Maîtrise des techniques de marketing produit : *benchmarking*, *road map*, rédaction d'un cahier des charges, spécifications fonctionnelles
- Maîtrise des outils informatiques courants (tableurs, bases de données...)
- Maîtrise des outils statistiques (*web analytics*) et de datamining

- Connaissance des règles éditoriales et d'ergonomie des sites web et/ou des applications mobiles
- Connaissances de base en architecture de site web et notions d'informatique de manière à pouvoir rédiger des *users story* (besoins utilisateurs) et spécifications fonctionnelles compatibles avec les possibilités de la technologie et dialoguer au quotidien avec les équipes de développement
- Maîtrise de la méthodologie agile / *scrum*

Aptitudes professionnelles

- Curiosité et goût pour l'investigation car il doit mener une veille sur les nouvelles tendances numériques et les nouveaux outils, et faire en permanence du *benchmarking*
- Écoute car il doit être capable d'être réceptif aux nouveaux modes de pensée et aux nouveaux usages des internautes
- Goût pour la technique et les innovations technologiques dans le domaine digital web et mobile pour proposer de nouveaux services
- Capacité d'adaptation, il exerce un rôle d'interface permanent et doit pouvoir adapter son discours et ses méthodes de travail en fonction de ses interlocuteurs
- Bonne expression orale car il est en contact avec des interlocuteurs variés au sein de l'entreprise et joue un rôle d'interface important
- Capacités d'analyse et de synthèse car il doit pouvoir effectuer l'analyse des actions menées mais aussi assurer les reportings et les bilans
- Force de proposition et créativité pour faire évoluer les types de produits et/ou services
- Réactivité, car il doit savoir réagir vite s'il découvre un nouveau service à développer ou à optimiser, de manière à ne pas se faire « doubler » par la concurrence
- Capacité à manager des équipes, à la fois hiérarchiquement et en mode projet

N°38: Ingénieur avant-vente

Autres intitulés :

- Ingénieur pré-sales
- Consultant avant-vente
- Ingénieur technico-commercial

Définition :

L'ingénieur avant-vente apporte un soutien technique à l'ingénieur commercial dans le cadre de propositions et de négociations commerciales.

Activités principales :

Veille technologique et économique

- Assurer une veille technologique sur l'ensemble des évolutions du marché.
- Réaliser un suivi économique et technologique des solutions concurrentes.
- Participer à des salons professionnels, conférences, aux diverses manifestations susceptibles d'enrichir sa connaissance du marché.

Développement de l'offre de l'entreprise

- Rendre crédible l'offre de l'entreprise en apportant son expertise sur le développement de nouveaux produits ou sur l'amélioration du produit existant.
- Formaliser les process de réponse aux appels d'offre.
- Initier de nouvelles idées en termes de présentation ou de démonstration auprès des clients.

Réponse aux appels d'offre ou propositions commerciales

- Comprendre les besoins du client et les enjeux du projet.
- Mener les études et les investigations complémentaires.
- Traduire le cahier des charges en spécifications pour le service technique.
- Maquetter une solution technologique pertinente en relation avec la direction technique.
- S'assurer de la faisabilité et de la rentabilité du projet pour l'entreprise.
- Élaborer la proposition technique.

Négociation commerciale

- Accompagner l'ingénieur commercial auprès du client afin d'apporter un soutien et une vision technique sur l'offre de l'entreprise.
- Effectuer des démonstrations techniques, des présentations produits auprès des décideurs et/ou utilisateurs.
- Répondre aux questions techniques complémentaires et rassurer le client dans sa prise de décision.

Profil :

Diplômes requis

- École d'ingénieurs (informatique, télécoms, généraliste)
- DESS / DEA informatique
- Diplôme de type Bac +4 en informatique : MIAGE, IUP informatique, maîtrise informatique, ingénieur maître...

Durée d'expérience

Ce poste s'adresse aux jeunes diplômés comme aux cadres confirmés en fonction de l'importance des projets gérés et de la complexité des compétences à mettre en œuvre

Compétences requises :

Compétences techniques

- Capacité d'adaptation aux différents environnements technologiques
- Connaissance pointue de son produit
- Expertise autour des technologies Web et des langages objets (J2EE, .Net, C++)
- Compétences fonctionnelles et organisationnelles
- Compréhension de la stratégie globale de l'entreprise et de son environnement concurrentiel
- Qualités rédactionnelles utiles pour la rédaction des appels d'offre
- Excellente culture technique permettant d'intégrer facilement l'ensemble des évolutions technologiques d'un marché
- L'anglais est incontournable car de nombreux appels d'offre et propositions sont rédigés en anglais

Aptitudes professionnelles

- Curiosité intellectuelle, afin de se tenir informé des évolutions technologiques du marché
- Sens commercial et aisance relationnelle dans le cadre des contacts avec les clients
- Qualités de rédaction pour formaliser les propositions envoyées aux clients
- Sens du travail en équipe, car il faut travailler en binôme avec un ingénieur commercial et souvent collaborer avec différents services en interne
- Force de décision et de conviction, pour arrêter un choix entre plusieurs solutions techniques et convaincre de la pertinence de ses propositions
- Forte résistance au stress, plus particulièrement dans le cadre des phases de négociation ou de réponse aux appels d'offre
- Implication personnelle car cette fonction est particulièrement exigeante en termes de flexibilité des horaires et de disponibilité, compte tenu du nombre de déplacements en France et à l'étranger

Partie 2 : notions par thèmes du numérique

Chapitre 1 : Système de l'information

1.1) Présentations :

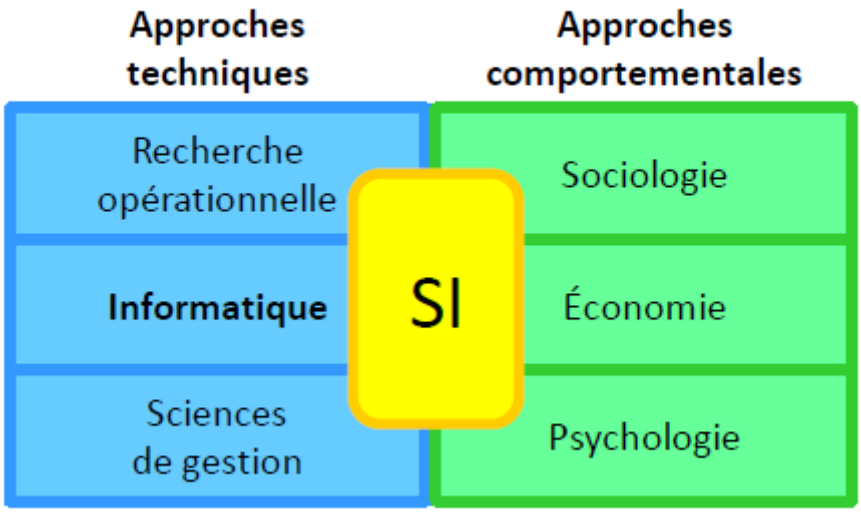
La société de l'information ou société de la connaissance est une société dans laquelle les technologies de l'information (TIC) jouent un rôle central.

Le Système de l'information (SI) peut être comparé à une sorte de système nerveux primaire de l'organisation. Dans une entreprise le SI est comme une circulation rapide d'une information de qualité entre les différentes « organes » ; délivrer la bonne information, au bon interlocuteur, au bon moment (-Prise de décisions appropriées, et action de l'entreprise adaptée à la situation) ; le SI contribue donc de manière évidente aux performances de l'organisation.

Une définition du *SI :*

Un système d'information est l'ensemble des ressources (matériels, logiciels, données, procédures, humains…) structurés pour acquérir, traiter, mémoriser, transmettre et rendre disponible l'information (sous forme de données, textes, sons, images,…) dans et entre les organisations.

Les systèmes de l'information peuvent être résumés par ce schéma :



1.2) Interaction entre les systèmes existants

Une organisation est composée d'un ensemble de systèmes. Il y a le système opérant, de pilotage et d'information, ce sont trois sous-systèmes qui interagissent entre eux :

- le **système opérant** : c'est ce qui est à la base de toute organisation, c'est ce système qui permet la transformation de l'information dont l'objectif est de la restituer à la bonne personne. Il correspond aux différents services d'une entreprise.

- le **système de pilotage** : C'est ce qui va contrôler et piloter le système opérant. Il se situe donc à la tête du système d'information fixant les objectifs et prenant les décisions.
- le **système d'information** : C'est ce qui intervient entre les deux autres systèmes. Ce système s'occupe de collecter, stocker, transformer et diffuser des données et informations dans le système opérant et de pilotage.

1.3) Les fonctions d'un système d'information

Il existe donc 4 fonctions principales d'un SI :

- **Collecter** : c'est à partir de là que naît la donnée, qu'on acquière les informations provenant de l'environnement interne ou externe à l'entreprise.
- **Stocker** : dès que l'information est acquise, le système d'information la conserve. Elle doit pouvoir être disponible et doit pouvoir être conservée dans le temps.
- **Transformer/traiter** : cette phase permet de transformer l'information et choisir le support adapté pour traiter l'information. Ici ont construit de nouvelles informations en modifiant le fond ou la forme.
- **Diffuser :** le SI transmet ensuite l'information dans son environnement interne ou externe.

L'objectif du SI est donc de restituer une information au sein d'une organisation directement exploitable par les différents acteurs et faciliter la prise de décision.

1.4) Le rôle du système d'information

Le système d'information et de décision permet de **stocker, traiter, restituer ou diffuser les informations** nécessaires au fonctionnement de l'organisation au moyen de l'informatique et des nouvelles technologies de l'intelligence collective (NTIC) comme les réseaux intranet, Internet…

1.5) Les ressources en systèmes d'information

Organiser le SI revient donc à définir les RESSOURCES utiles au bon fonctionnement de l'organisation.

Je fais une petite liste des ressources :

- ***Ressources humaines*** : Les utilisateurs sont les premiers acteurs du système d'information. Ils génèrent de l'information.

Les spécialistes informatiques sont également utiles pour faire fonctionner le système, le maintenir et le faire évoluer.

- ***Procédures / Processus*** : Une **procédure** dans le domaine de l'entreprise et des affaires désigne une manière spécifiée d'effectuer un ensemble de tâches. Elle représente la mise en œuvre de tout ou d'une partie d'un processus et est destinée à être reproductible. Elle décrit ainsi étape par étape l'enchainement des tâches à réaliser, et les rôles et responsabilités associées.
- ***Ressources logicielles*** : Les programmes, applications et services permettent de réaliser un certain nombre de traitements de l'information. Chaque logiciel (en anglais « software ») est un ensemble de séquences d'instructions interprétables par une machine et qui détermine donc les tâches qui peuvent-être effectuées par la machine, ordonne son fonctionnement et lui procure ainsi son utilité fonctionnelle.

- ***Données / Connaissance :*** Une **donnée** est ce qui est connu et qui sert de point de départ à un raisonnement ayant pour objet la détermination d'une solution à un problème en relation avec cette donnée. Cela peut être une description élémentaire d'une réalité, le résultat d'une comparaison entre deux événements du même ordre (mesure) soit en d'autres termes une observation ou une mesure.
- ***Ressources matérielles*** : Le matériel désigne tout d'abord les équipements dont le **matériel informatique** (en anglais « *hardware* ») : c'est l'ensemble des pièces détachées des appareils informatiques. Il y a des pièces situées à l'intérieur du boîtier de l'ordinateur aussi bien qu'à l'extérieur (les périphériques).

1.6) Le système de l'information dans l'informatique

Le système de l'information est regroupé en plusieurs thèmes :

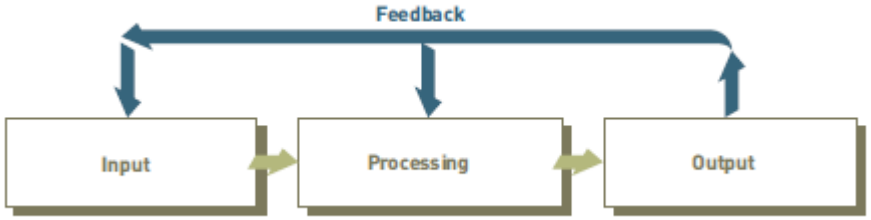
- Personnelles : se sont toutes les personnes qui travaille pour le numérique, ingénieur(e)s, chef(fe) de projets, etc.
- Matériel : (hardware) est la couche technique et électronique par exemple les serveurs.
- Logiciel : (software) est le développement outil numérique en web et applications.
- Télécommunication : (Telecommunication) se sont touts les moyens de communication : Internet, 4G etc.
- Base de données : (Databases) sont les moyens de stockage.
- Procédures : se sont tout les principes pour réaliser une tâche.

On représente le SI organisation suivante :

- Bases de données de l'entreprise
- Prologiciel de gestion intégré (ERP, Entreprise Ressource Planning)
- Outil de gestion de la relation Client (CRM, Customer Relationship Management)
- Outil de gestion de la chaîne logistique (SCM, Supply Chain Management)
- Outil d'informatique décisionnelle (Business Intelligence)
- Applications métiers (sur site et mobile)
- Infrastructure réseau
- Serveurs de données et système de stockage
- Serveur d'applications (ou middleware)
- Dispositifs de sécurité.

1.7) Qu'est-ce qu'un système informatique :

Un SI est résumé par ce schéma :



Input : Une entrée est de l'activité.

Transformation: Conversion ou transformation de données en sorties utiles.

Sortie : Production d'informations utiles, généralement sous forme de documents et rapports.

Retour d'information (feedback) : Sortie utilisée pour apporter des modifications aux activités de saisie ou de traitement.

Chapitre 2 : e-entreprise

2.1 Définitions

Une **entreprise** peut être comme une entité fournissant des produits ou services à des clients, en s'appuyant sur les produits ou services de partenaires dans un environnement en constante évolution.

- Les **fonctions de réalisation** qui représentent le cœur de son activité, c'est-à-dire la production de biens ou de services. Elles concernent les activités de production, de gestion de stocks et de l'approvisionnement (fonction achat).
- Les **fonctions de management** qui regroupent toutes les fonctions stratégiques de gestion de l'entreprise. Elles regroupent la direction générale de l'entreprise, les fonctions de gestion des ressources humaines (RH), ainsi que les fonctions de gestion financière et comptable.
- Les **fonctions support** qui servent d'appui aux fonctions de réalisation pour permettre le bon fonctionnement de l'entreprise. Il s'agit de l'ensemble des activités liées à la vente (dans certains cas elles font partie du cœur du métier), ainsi que l'ensemble des activités transversales à l'organisation, telle que la gestion des infrastructures technologiques (fonction IT).

2.2 L'e-commerce

Les entreprises sont généralement caractérisées par le type de relations commerciales qu'elles entretiennent.

- **B to B** (Business to Business, parfois noté B2B) qui désigne une relation commerciale d'entreprise à une entreprise et qui est basée sur l'utilisation d'un support numérique pour les échanges d'information.
- **B to C** (Business to Consumer, parfois noté B2C) qui désigne une relation commerciale entre une entreprise et le grand public (particuliers). Il s'agit donc de ce qu'on appelle le **commerce électronique**, dont la définition ne se limite pas au seul acte de vente, mais couvre également tous les échanges qu'une entreprise peut avoir avec ces clients (de la demande de devis au service après-vente).
- **B to A** (Business to Administration, parfois noté B2A) qui désigne une relation entre une entreprise et le secteur public (administration fiscale, etc.) et qui s'appuie sur des mécanismes d'échange numérique (téléprocédures, formulaires électroniques…).

2.3 Front Office/Back Office

- Le **Front Office** (parfois appelé également Front Line) désigne la partie frontale de l'entreprise, visible par la clientèle.
- Le **Back Office** à l'inverse désigne l'ensemble des parties du système d'information auxquelles l'utilisateur final n'a pas accès.

Chapitre 3 : client/serveur

Les standards du Web sont basés sur un mode de fonctionnement dit client/serveur où un utilisateur interagit avec un service par l'intermédiaire de requêtes.

3.1 Architecture client/serveur

Les applications Web fonctionnent selon un environnement client/serveur, cela signifie que des machines clientes (des machines faisant partie du réseau) contractent un serveur, une machine généralement très puissante en termes de capacités d'entrée-sortie, qui leur fournit des services. Ces services sont des programmes fournissant des données telles que l'heure, la fiche d'un produit, un fichier, etc.

Les services sont exploités par des programmes, appelés programmes clients, s'exécutant sur les machines clientes.

On parle ainsi de client (client Web, client FTP, client de messagerie, etc.) lorsque l'on désigne un programme tournant sur une machine cliente, capable de traiter des informations qu'il récupère auprès d'un serveur (dans le cas du client Web il s'agit de pages Web, tandis que pour le client de messagerie il s'agit de courrier électronique).

3.2 Architecture mainframe

Les premiers réseaux informatiques étaient architecturés autour d'un ordinateur central, appelé mainframe.

Le mainframe représente ainsi un ordinateur central de grande puissance chargé de gérer les sessions utilisateurs des différents terminaux qui lui étaient reliées.

3.3 Architectures multiniveaux

a. Architecture à 2 niveaux

L'architecture à deux niveaux (aussi appelée architecture 2 tier) caractérise les systèmes client/serveur pour lesquels le client demande une ressource et le serveur la lui fournit directement, en utilisant ses propres ressources. Cela signifie que le serveur ne fait pas appel à une autre application afin de fournir une partie du service.

b. Architecture à 3 niveaux

Dans *l'architecture à 3 niveaux* (appelée architecture 3-tier) existe un niveau intermédiaire, c'est-à-dire que l'on a généralement une architecture partagée entre :

- Un *client*, c'est-à-dire l'ordinateur demandeur de ressources équipées d'une interface utilisateur (généralement un navigateur Web) chargée de la présentation.
- Le *serveur d'applications* (appelé également middleware) chargé de fournir la ressource mais faisant appel à un autre serveur.
- Le *serveur de données* fournissant au serveur d'application les données dont il a besoin.

3.4 Type de clients

a. Client léger

Le *terme client léger* (parfois client pauvre, thin client) désigne une application accessible via une interface Web (en HTML pour HyperText Markup Language) consultable à l'aide d'un navigateur Web, où la totalité de la logique métier est traitée du côté du serveur.

Pour ces raisons, le navigateur est parfois appelé client universel.

b. Client lourd

Le *terme client lourd* (fat client ou heavy client), par opposition au client léger, désigne une application cliente graphique exécutée sur le système d'exploitation de l'utilisateur. Un client lourd possède généralement des capacités de traitement évoluées et peut posséder une interface graphique sophistiquée. Néanmoins, ceci demande un effort de développement et tend à mêler la logique de présentation (l'interface graphique) avec la logique applicative (les traitements).

Ce type d'applications étant généralement installé sur le système d'exploitation de l'utilisateur, une nouvelle version doit être installée afin de la faire évoluer. Pour y remédier, les éditeurs d'applications lourdes les dotent généralement d'une fonctionnalité exécutée au lancement de l'application, permettant de vérifier sur un serveur distant si une version plus récente est disponible et le cas échéant propose à l'utilisateur de la télécharger et de l'installer.

c. Client riche

Un client riche est compromis entre le client léger et le client lourd. L'objectif du client riche est donc de proposer une interface graphique, décrite avec une grammaire de description basée sur la syntaxe XML, permettant d'obtenir des fonctionnalités similaires celles d'un client lourd (glisser-déposer, onglets, multifenêtrage, menus déroulants, etc).

Les clients riches permettent ainsi de gérer l'essentiel des traitements du côté serveur. Les données sont ensuite transmises dans un format d'échange standard via les services Web utilisant la syntaxe (SOAP, XML-RPC), puis interprétées par le client riche.

Les clients riches sont souvent intégrés aux navigateurs sous forme de plug-ins ou indépendamment du navigateur.

Chapitre 4 : les réseaux

Dans ce chapitre 4 sur les réseaux, je vais m'intéresser sur l'essentiel que j'ai pensé utile de connaître pour un ingénieur développeur informatique.

4.1 Protocoles réseaux

4.1.1 Notion de protocole

Un protocole est une méthode standard qui permet la communication entre des processus (s'exécutant éventuellement sur différentes machines), c'est-à-dire un ensemble de règles et de procédures à respecter pour émettre et recevoir des données sur un réseau. Il en existe plusieurs selon que l'on attend de la communication. Certains protocoles seront par exemple spécialisés dans l'échange de fichiers (le FTP), d'autres pourront servir à gérer simplement l'état de la transmission et des erreurs (c'est le cas du protocole ICMP)…

Sur Internet, les protocoles utilisés font partie d'une suite de protocoles, c'est-à-dire un ensemble de protocoles reliés entre eux. Cette suite de protocole s'appelle TCP/IP. Elle contient, entre autres, les protocoles suivants : http, ftp, ARP, ICMP, IP, TCP, UDP, SMTP, Telnet, NNTP.

On classe généralement les protocoles en deux catégories selon le niveau de contrôle que l'on désire :

- Les *protocoles orientés connexion* : il s'agit des protocoles opérant un contrôle de transmission des données pendant une communication établie entre deux machines. Dans un tel schéma, la machine réceptrice envoie des accusés de réception lors de la communication, ainsi la machine émettrice est garantie de la validité des données qu'elle envoie. Les données sont ainsi envoyées sous forme de flot. Ex : TCP est un protocole orienté connexion.
- Les *protocoles non orientés connexion* : il s'agit d'un mode de communication dans lequel la machine émettrice envoie des données sans prévenir la machine réceptrice, et la machine réceptrice reçoit les données sans envoyer d'avis de réception à la première. Les données sont ainsi envoyées sous forme de blocs (datagrammes). Ex : UDP est un protocole non orienté connexion.

4.1.2 Adresse IP

Sur Internet, les ordinateurs communiquent entre eux grâce au protocole IP (Internet Protocol), qui utilise des adresses numériques, appelées adresse IP. C'est l'ICANN (Internet Corporation for Assigned Names and Numbers, depuis 1998) qui est chargée d'attribuer des adresses IP publiques, c'est-à-dire les adresses IP des ordinateurs directement connectés sur le réseau public Internet.

Chaque ordinateur a une adresse IP pour communiquer entre eux.

4.2 Modèles de référence

4.2.1 Le modèle de référence OSI

Le modèle OSI (moins le support physique) est illustré sur la figure 1.

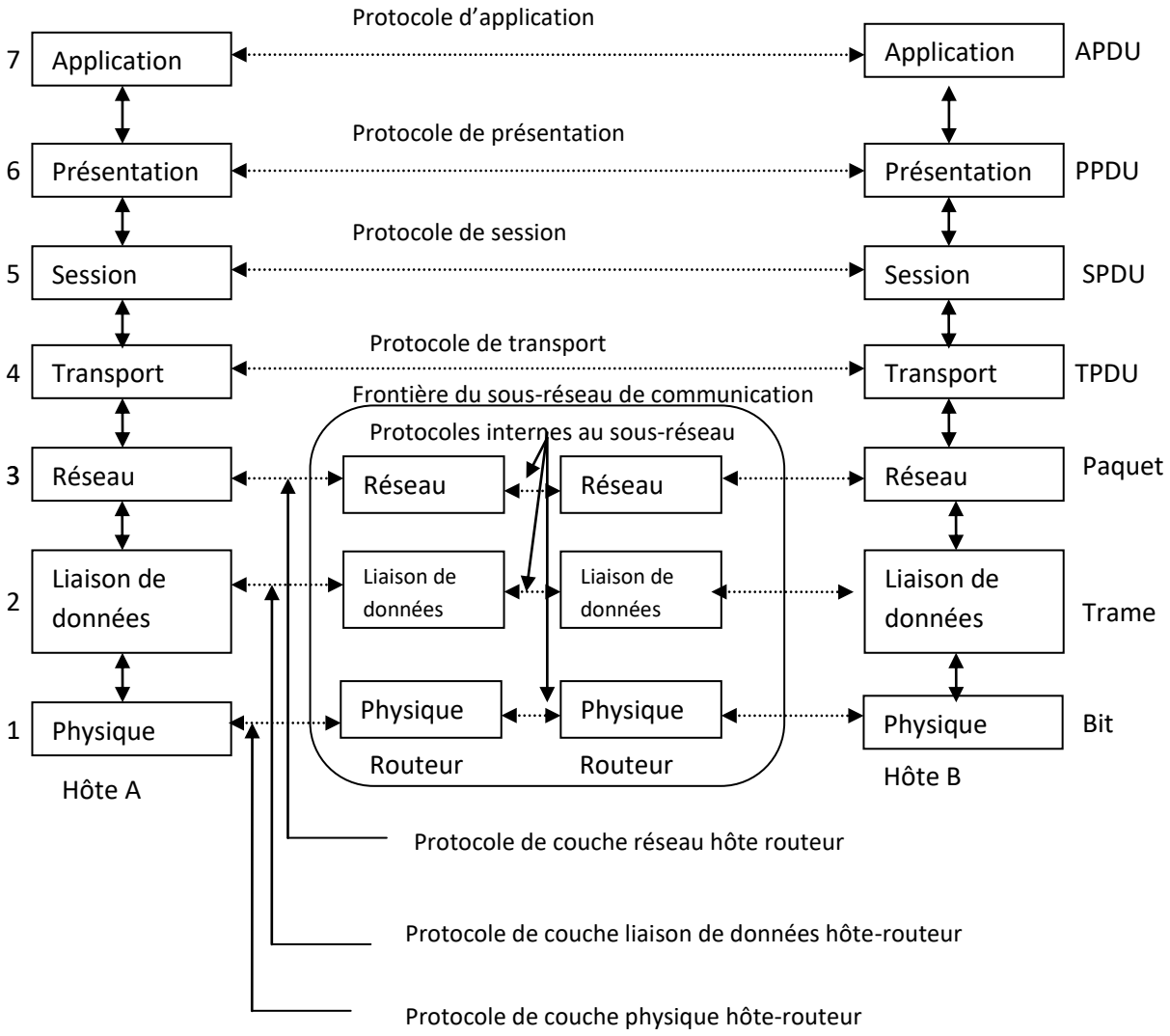


Figure 1: Le modèle de référence OSI

Il s'appuie sur une proposition qui a été développé par l'ISO (Organisation internationale de normalisation) comme une première étape vers la normalisation internationale des protocoles utilisés dans les diverses couches.

On l'appelle modèle de référence OSI (Open Systems Interconnection) car il traite des systèmes ouverts, c'est-à dire des systèmes ouverts à la communication avec d'autres systèmes.

Ce modèle est composé de 7 couches. Les principes qui ont été appliqués pour parvenir à ce nombre peuvent être résumés brièvement de la façon suivante :

1. Une couche doit être créée lorsqu'un nouveau niveau d'abstraction est nécessaire.
2. Chaque couche doit assurer une fonction bien définie.
3. La fonction de chaque couche doit être choisie en visant la définition de protocoles normalisés au niveau internationale.
4. Les limites d'une couche doivent être fixées de manière à réduire la quantité d'informations devant passer au travers des interfaces.
5. Le nombre de couches doit être, d'une part, assez grand pour que des fonctions très distinctes ne soient pas regroupées dans une même couche et, d'autre part, suffisamment faible pour que l'architecture ne devienne pas trop complexe.

Je vais maintenant expliquer chaque couche du modèle OSI :

- **La couche physique** :

La couche physique se charge de la transmission de bits à l'état brut sur un canal de communication. L'un des objectifs de conception de ce niveau est de s'assurer qu'un bit à 1 envoyé sur une extrémité arrive aussi à 1 de l'autre côté et non à 0.

- **La couche liaison de données** :

La couche liaison de données définit l'interface avec la carte de réseau et le partage du média de transmission.

- **La couche réseau** :

La couche réseau permet de gérer l'adressage et le routage des données, c'est-à-dire leur acheminement via le réseau.

- **La couche transport** :

La couche transport est chargée du transport des données, de leur découpage en paquets et de la gestion des éventuels les erreurs de transmission.

- **La couche session** :

La couche session définit l'ouverture et la destruction des sessions de communication entre les machines du réseau.

- **La couche présentation** :

La couche présentation définit le format des données manipulées par le niveau applicatif (leur représentation, éventuellement leur compression et leur chiffrement) indépendamment du système.

- **La couche application** :

La couche application assure l'interface avec les applications. Il s'agit donc du niveau le plus proche des utilisateurs, géré directement par les logiciels.

Cette couche contient différents protocoles dont les utilisateurs ont couramment besoin. http qui forme la base du World Wide Web, est un protocole d'application largement utilisé. Lorsqu'un navigateur veut afficher une page Web, il transmet son nom au serveur qui l'hébergeur au moyen du protocole http, et le serveur envoie la page en réponse.

4.2.2 Le modèle de référence TCP/IP

Le modèle TCP/IP est illustré sur la figure 2.

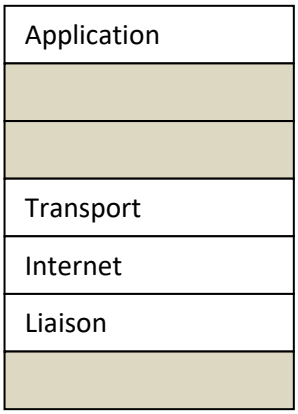


Figure 2: Le modèle TCP/IP

On peut résumer dans ce tableau suivant :

Niveau	Modèle TCP/IP	Modèle OSI	Protocoles TCP/IP
Niveau 4	Couche application	Couche Application Couche Présentation Couche Session	Applications réseau (Telnet, SMTP, FTP,…)
Niveau 3	Couche Transport (TCP)	Couche Transport	TCP ou UDP
Niveau 2	Couche Internet (IP)	Couche Réseau	IP, ARP, RARP
Niveau 1	Couche Accès réseau	Couche Liaison données Couche Physique	FTS, FDDI, PPP, Ethernet, Anneau à jeton (Token Ring)

- La **couche Accès réseau** spécifie la forme sous laquelle les données doivent être acheminées quel que soit le type de réseau utilisé.
- La **couche Internet** est chargée de fournir le paquet de données (datagramme).
- La **couche Transport** assure l'acheminement des données, ainsi que les mécanismes permettant de connaître l'état de la transmission.
- La **couche Application** englobe les applications standards du réseau.

4.3 Les familles de réseaux

Je liste les réseaux existants :

- Les réseaux IP
- Les réseaux SDN : Software Defined Networking
- Le cloud Networking
- Les réseaux open source

- Les réseaux d’accès terrestres
- Les réseaux d’accès hertziens
- Les scall cells
- Les réseaux multisaut
- Les réseaux de mobiles 1G à 5G
- Les réseaux sans fils
- Les réseaux Wi-Fi

Chapitre 5 : les protocoles applicatifs

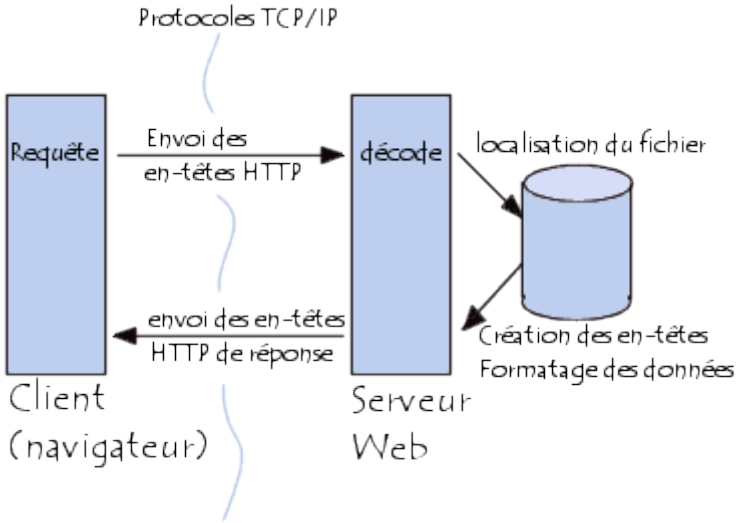
5.1 Protocole http

Le protocole http signifie HyperText Transfert Protocol.

Communication entre navigateur et serveur

La communication entre le navigateur et le serveur se fait en deux temps :

- Le navigateur effectue une requête http
- Le serveur traite la requête puis envoie une réponse http



En réalité la communication s'effectue en plus de temps si on considère le traitement de la requête par le serveur. Etant donné que l'on s'intéresse uniquement au protocole HTTP, le traitement du côté serveur ne sera pas explicité dans le cadre de cet article... Si ce sujet vous intéresse, référez-vous à l'article sur le traitement des CGI.

Requête http

Une requête HTTP est un ensemble de lignes envoyé au serveur par le navigateur. Elle comprend :

- **Une ligne de requête**: c'est une ligne précisant le type de document demandé, la méthode qui doit être appliquée, et la version du protocole utilisée. La ligne comprend trois éléments devant être séparés par un espace :
  - La méthode
  - L'URL
  - La version du protocole utilisé par le client (généralement *HTTP/1.0*)
- **Les champs d'en-tête de la requête**: il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la requête et/ou le client (Navigateur, système d'exploitation, ...). Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête
- **Le corps de la requête**: c'est un ensemble de lignes optionnelles devant être séparées des lignes précédentes par une ligne vide et permettant par exemple un envoi de données par une commande POST lors de l'envoi de données au serveur par un formulaire.

Une requête HTTP a donc la syntaxe suivante (*<crlf>* signifie retour chariot ou saut de ligne) :

METHODE URL VERSION*<crlf>*
EN-TETE : Valeur*<crlf>*
.
.
.
EN-TETE : Valeur*<crlf>*
Ligne vide<crlf>
CORPS DE LA REQUETE

- **Commandes**

Commande	**Description**
GET	Requête de la ressource située à l'URL spécifiée
HEAD	Requête de l'en-tête de la ressource située à l'URL spécifiée
POST	Envoi de données au programme situé à l'URL spécifiée
PUT	Envoi de données à l'URL spécifiée
DELETE	Suppression de la ressource située à l'URL spécifiée

- **En-têtes**

Nom de l'en-tête	**Description**
Accept	Type de contenu accepté par le browser (par exemple *text/html*). Voir types MIME
Accept-Charset	Jeu de caractères attendu par le browser
Accept-Encoding	Codage de données accepté par le browser
Accept-Language	Langage attendu par le browser (anglais par défaut)
Authorization	Identification du browser auprès du serveur
Content-Encoding	Type de codage du corps de la requête
Content-Language	Type de langage du corps de la requête
Content-Length	Longueur du corps de la requête

Content-Type	Type de contenu du corps de la requête (par exemple *text/html*). Voir types MIME
Date	Date de début de transfert des données
Forwarded	Utilisé par les machines intermédiaires entre le browser et le serveur
From	Permet de spécifier l'adresse e-mail du client
From	Permet de spécifier que le document doit être envoyé s'il a été modifié depuis une certaine date
Link	Relation entre deux URL
Orig-URL	URL d'origine de la requête
Referer	URL du lien à partir duquel la requête a été effectuée
User-Agent	Chaîne donnant des informations sur le client, comme le nom et la version du navigateur, du système d'exploitation

- **Réponse HTTP**

Une réponse HTTP est un ensemble de lignes envoyées au navigateur par le serveur. Elle comprend :

- **Une ligne de statut**: c'est une ligne précisant la version du protocole utilisé et l'état du traitement de la requête à l'aide d'un code et d'un texte explicatif. La ligne comprend trois éléments devant être séparés par un espace :
  - La version du protocole utilisé
  - Le code de statut
  - La signification du code
- **Les champs d'en-tête de la réponse**: il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la réponse et/ou le serveur. Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête
- **Le corps de la réponse**: il contient le document demandé

Une réponse HTTP a donc la syntaxe suivante (*<crlf>* signifie retour chariot ou saut de ligne) :

```
ERSION-HTTP CODE EXPLICATION<crlf>
EN-TETE : Valeur<crlf>
.
.
.
```

```
EN-TETE : Valeur<crlf>
Ligne vide<crlf>
CORPS DE LA REPONSE
```

Voici donc un exemple de réponse HTTP :

```
HTTP/1.0 200 OK
Date : Sat, 15 Jan 2000 14:37:12 GMT Server : Microsoft-IIS/2.0
Content-Type : text/HTML
Content-Length : 1245
Last-Modified : Fri, 14 Jan 2000 08:25:13 GMT
```

- **En-têtes de réponse**

Nom de l'en-tête	Description
Content-Encoding	Type de codage du corps de la réponse
Content-Language	Type de langage du corps de la réponse
Content-Length	Longueur du corps de la réponse
Content-Type	Type de contenu du corps de la réponse (par exemple *text/html*). Voir types MIME
Date	Date de début de transfert des données
Expires	Date limite de consommation des données
Forwarded	Utilisé par les machines intermédiaires entre le browser et le serveur
Location	Redirection vers une nouvelle URL associée au document
Server	Caractéristiques du serveur ayant envoyé la réponse

- **Les codes de réponse**

Ce sont les codes que vous voyez lorsque le navigateur n'arrive pas à vous fournir la page demandée. Le code de réponse est constitué de trois chiffres : le premier indique la classe de statut et les suivants la nature exacte de l'erreur.

Code	Message	Description
10x	**Message d'information**	**Ces codes ne sont pas utilisés dans la version 1.0 du protocole**
20x	**Réussite**	**Ces codes indiquent le bon déroulement de la transaction**
200	OK	La requête a été accomplie correctement
201	CREATED	Elle suit une commande POST, elle indique la réussite, le corps du reste du document est sensé indiquer l'URL à laquelle le document nouvellement créé devrait se trouver.
202	ACCEPTED	La requête a été acceptée, mais la procédure qui suit n'a pas été accomplie
203	PARTIAL INFORMATION	Lorsque ce code est reçu en réponse à une commande GET, cela indique que la réponse n'est pas complète.
204	NO RESPONSE	Le serveur a reçu la requête mais il n'y a pas d'information à renvoyer
205	RESET CONTENT	Le serveur indique au navigateur de supprimer le contenu des champs d'un formulaire
206	PARTIAL CONTENT	Il s'agit d'une réponse à une requête comportant l'en-tête *range*. Le serveur doit indiquer l'en-tête *content-Range*
30x	**Redirection**	**Ces codes indiquent que la ressource n'est plus à l'emplacement indiqué**
301	MOVED	Les données demandées ont été transférées à une nouvelle adresse
302	FOUND	Les données demandées sont à une nouvelle URL, mais ont cependant peut-être été déplacées depuis...
303	METHOD	Cela implique que le client doit essayer une nouvelle adresse, en essayant de préférence une autre méthode que GET
304	NOT MODIFIED	Si le client a effectué une commande GET conditionnelle (en demandant si le document a été modifié depuis la dernière fois) et que le document n'a pas été modifié il renvoie ce code.
40x	**Erreur due au client**	**Ces codes indiquent que la requête est incorrecte**
400	BAD REQUEST	La syntaxe de la requête est mal formulée ou est impossible à

		satisfaire
401	UNAUTHORIZED	Le paramètre du message donne les spécifications des formes d'autorisation acceptables. Le client doit reformuler sa requête avec les bonnes données d'autorisation
402	PAYMENT REQUIRED	Le client doit reformuler sa demande avec les bonnes données de paiement
403	FORBIDDEN	L'accès à la ressource est tout simplement interdit
404	NOT FOUND	Classique! Le serveur n'a rien trouvé à l'adresse spécifiée. Parti sans laisser d'adresse... :)
50x	**Erreur due au serveur**	**Ces codes indiquent qu'il y a eu une erreur interne du serveur**
500	INTERNAL ERROR	Le serveur a rencontré une condition inattendue qui l'a empêché de donner suite à la demande (comme quoi il leur en arrive des trucs aux serveurs...)
501	NOT IMPLEMENTED	Le serveur ne supporte pas le service demandé (on ne peut pas tout savoir faire...)
502	BAD GATEWAY	Le serveur a reçu une réponse invalide de la part du serveur auquel il essayait d'accéder en agissant comme une passerelle ou un proxy
503	SERVICE UNAVAILABLE	Le serveur ne peut pas vous répondre à l'instant présent, car le trafic est trop dense (toutes les lignes de votre correspondant sont occupées veuillez rappeler ultérieurement)
504	GATEWAY TIMEOUT	La réponse du serveur a été trop longue vis-à-vis du temps pendant lequel la passerelle était préparée à l'attendre (le temps qui

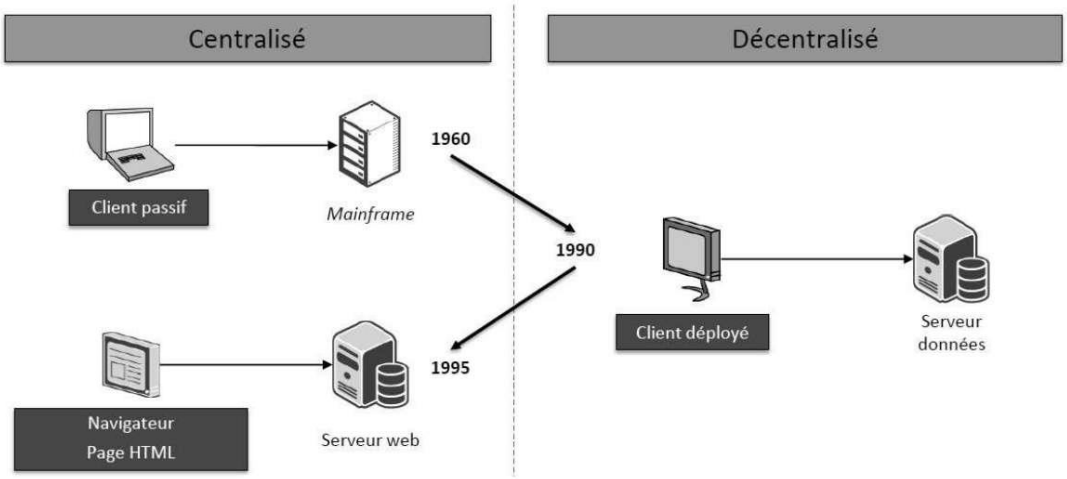
Chapitre 6 : cloud computing

6.1 Le concepts du cloud computing

Depuis sa montée en puissance dans les années 1960, l'architecture informatique suit un cycle régulier de centralisation/décentralisation. Ainsi les premiers systèmes utilisés en entreprises étaient des mainframes, c'est-à-dire des machines dans lesquelles toute la logique de calcul et de persistance de l'information était centralisée. De nombreuses évolutions sont passés par le Minitel et les terminaux passifs étaient composés d'un simple couple écran/clavier et constituaient des interfaces d'accès interchangeables, qui ne contenaient aucune donnée utilisateur.

Au début des années 1990, sont apparues les architectures client/serveur qui ont permis le report des traitements sur les postes de travail, les fameux ordinateurs personnels (PC ou Personal Computer), inventés par IBM. Ces PC ont permis la montée en puissance de Microsoft qui leur à fourni leurs logiciels embarqués : les incontournables Windows et Office.

Au milieu de ces mêmes années 1990, les architectures web ont conduit à la recentralisation de la logique de traitement sur des serveurs centraux, ramenant le PC à un simple dispositif d'affichage au travers du navigateur. Elles ont permis l'usage d'applications à échelle de l'Internet grâce aux standards http et HTML. De plus, elles ont permis un accès aux applications sans passer par la douloureuse phase de déploiement logiciel sur chacun des PC du parc informatique.



6.2 La montée en puissance du web

Les standards web (http et HTML) ont été inventés en 1990 par Tim Berners-Lee. Son idée est de faire une encyclopédie à la manière de Wikipédia. Puis à la fin des années 1990, ces sites, au départ statiques, ont commencé à devenir transactionnels, permettant l'émergence du commerce électronique, pour devenir de véritables applications informatiques.

Le web a aussi introduit un changement dans l'évolution de l'informatique : en effet, des innovations ont commencé à être testées auprès du grand public.

6.3 Le WEB 2.0

Le terme web 2.0 a été créé en 2005 par Tim O'Reilly, selon O'Reilly, le web 2.0 consiste à considérer le web comme une plateforme.

C'est l'ère du Cloud Computing.

6.4 L'origine du terme Cloud Computing

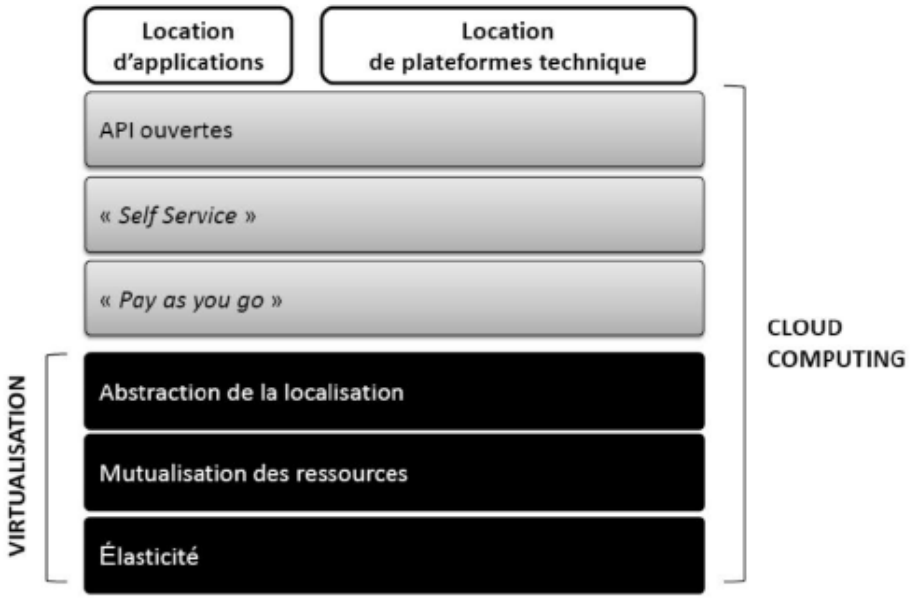
Le terme cloud computing est traduit littéralement par « informatique dans les nuages », ces nuages faisant référence à Internet et au Web.

Sur un plan plus technique, on peut considérer que le cloud computing est une évolution des technologies de virtualisation. La virtualisation permet de donner plus d'agilité aux centres de données, grâce aux trois propriétés suivantes :

- **Mutualisation des ressources** : la virtualisation permet d'affecter les ressources d'une même machine à plusieurs applications ;
- **Abstraction sur la localisation** : l'application est « quelque part » sur l'une des machines constitutives de la plateforme de virtualisation. Si cette plateforme utilise des mécanismes de réplication sur des Datacenters distants, les risques de désastre (incendies, inondations) sont couverts par la distribution multisites ;
- **Elasticité** : il est possible d'allouer des ressources supplémentaires à une application proche de la saturation, dans les limites physiques de la plateforme dispose de grandes ressources de puissance inutilisées, on peut affecter en quelques instants des capacités supplémentaires à une application.

Le cloud computing ajoute d'autres propriétés à celles de la virtualisation :

- Le **Pay As You Go** : les utilisateurs paient les ressources qu'ils utilisent en fonction de leur consommation réelle et précise.
- Le **Self Service** : l'équipe de développement peut demander l'allocation de ressources via un portail web. Ces ressources seront mises à sa disposition de manière automatique quelques minutes plus tard ;
- Les **API ouvertes** : les plateformes cloud proposent des interfaces techniques accessibles à distance qui permettent de les intégrer avec le système d'information ou bien de piloter les services à distance.



Le concept de cloud computing englobe les concepts de Sofware as a Service (SaaS), de Platform as aa Service (PaaS), et d'Infrastructure as a Service (IaaS).

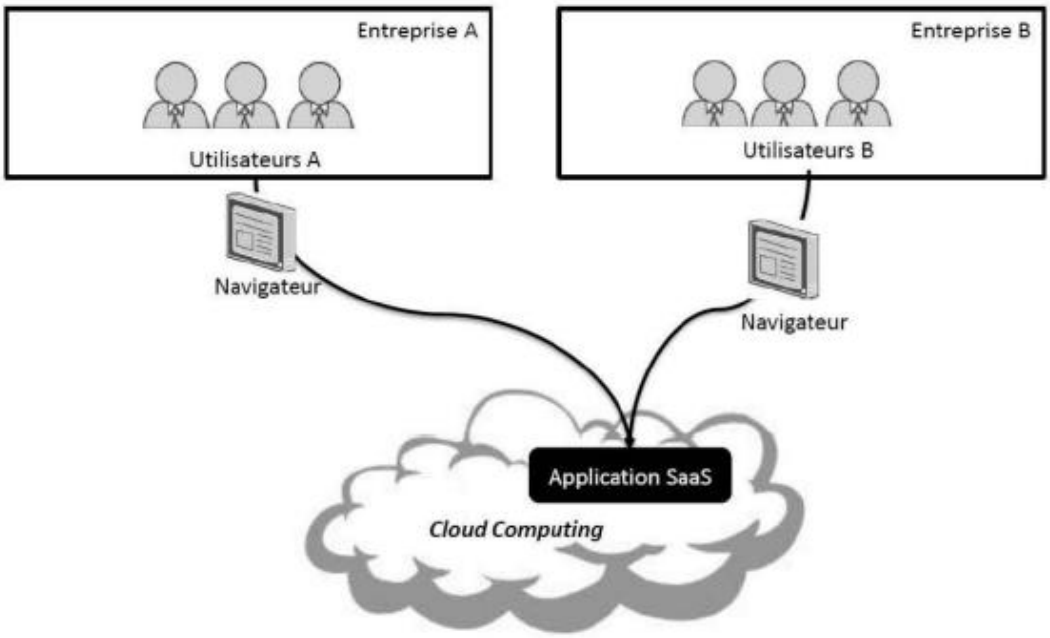
Le terme as as Service évoque bien un service, dans le sens où le fournisseur vend une fonction opérationnelle, et non des composants techniques nécessitant une compétence informatique.

6.5 Que signifie SAAS ?

SaaS signifie Software as a Service, c'est-à-dire un logiciel fourni sous la forme de service. Il s'agit donc de location d'application opérationnelle, clef en main, et non d'achat de logiciel informatique, à installer soi-même sur une machine. Les SaaS s'adressent donc aux utilisateurs finaux.

La différence entre SaaS et logiciel est essentielle. En effet, les SaaS proposent des logiciels opérationnels, prêts à l'emploi, sans passer par une étape d'installation, et sans aucune tâche de maintenance.

Les SaaS sont exécutés sur des plateformes conçues pour une utilisation simultanée par un grand nombre de collaborateurs qui travaillent dans de nombreuses entreprises différentes. Ces plateformes sont mises à disposition par des acteurs (comme Google ou Salesforce) que j'appelle opérateurs SaaS, car leur métier est plus proche de ceux des opérateurs télécoms que celui des éditeurs de logiciel.



Le canal d'accès aux applications SaaS étant le réseau Internet, on les appelle aussi « applications en ligne ».

6.6 Que signifie PAAS ?

PaaS signifie Platform as a Service ou plateforme sous forme de service. Il s'agit de location de plateforme technique, permettant l'exécution de code développé en spécifique. Les PaaS s'adressent donc aux développeurs.

Le PaaS désigne une plateforme d'exécution hébergée par un opérateur et accédée depuis Internet. Cette plateforme peut être utilisée pour exécuter des sites Web, des SaaS, ou tout développement spécifique issu de l'entreprise.

6.7 Que signifie IAAS ?

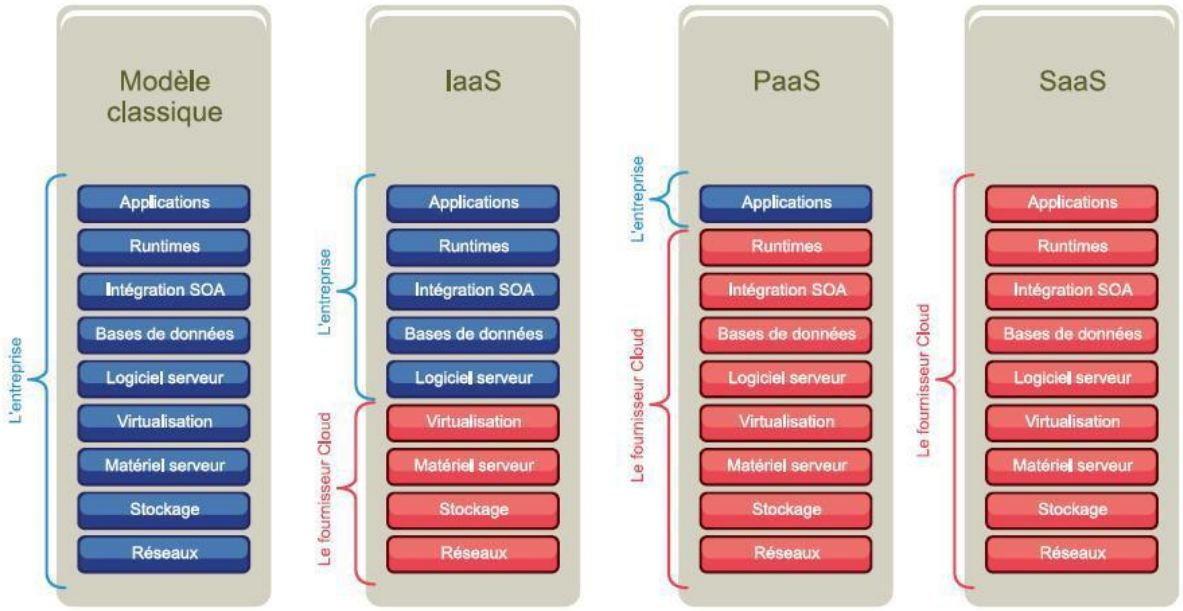
IaaS signifie Infrastructure as a Service ou infrastructure sous forme de service. Il s'agit de location de plateforme technique, permettant l'exécution d'architectures applicatives complètes, comprenant base de données, serveur d'application, etc. LessIaaS s'adressent donc aux équipes d'exploitation.

Le IaaS désigne une plateforme d'hébergement exploitée par un opérateur et accessible depuis Internet.

Cette plateforme peut être utilisée pour exécuter des SaaS.

6.8 Résumé

IaaS, PaaS, SaaS : qui maintient quoi ?



Modèle IaaS : Infrastructure as a Service désigne qu'on dispose d'une infrastructure (serveurs, stockage, réseau) hébergée. L'accès à la ressource est complet et sans restriction, équivalent de fait à la mise à disposition d'une infrastructure physique réelle.

Modèle PaaS : Plateform as a Service désigne qu'on dispose d'une plateforme capable d'accueillir les applications de l'entreprise et tous les environnements et outils de gestion et de test. L'environnement est prêt à l'emploi, fonctionnel et performant, y compris en production ; l'infrastructure hébergée étant totalement transparente.

Modèle SaaS : Software as a Service désigne la déportation des applications de l'entreprise dans le cloud. Les Cloud Computing SaaS proposent des logiciels opérationnels prêts à l'emploi sans aucune installation ou opération de maintenance.

Les applications d'entreprise concernée par le type de fonctionnement : CRM, outils collaboratifs, messagerie, BI, ERP…

Trois types de Cloud Computing :

- Cloud public
- Cloud privé
- Cloud hybride

Cloud public : externe à l'entreprise et partagé entre plusieurs entités, accès via internet, payement de type « pay as-you-go manner» pour le grand public ou abonnement pour les entreprises. Il est géré par un prestataire externe propriétaire des infrastructures, avec des ressources partagée entre plusieurs sociétés (ou ouvertes au grand public).

Cloud privé : structure interne à l'entreprise ou à un groupement d'entreprises ou cloud externe et complément dédié en accès sécurisé sur internet mutualisé entre les différentes entités d'une seule et même entreprise. Le Cloud communautaire est un cas particulier dont le cloud privé est ouverts aux partenaires de l'entreprise : clients, fournisseurs, institutions financières, Be, etc…

Cloud hybride : conjonction de deux types.

Chapitre 7 : Les architectures du cloud computing

7.1 Cloud Computing et Architectures Muti-Tiers

Les architectures muti-tiers constituent l'état des architectures d'entreprise. Elles sont utilisées dans les systèmes informatiques depuis environ 15 ans. Cette pratique consiste à segmenter les couches techniques des applications en plusieurs « tiers » conçu et exploités de manière autonome. On considère généralement les tiers suivants :

- Le **serveur de présentation**, dont le rôle est de produire les écrans visibles par les utilisateurs. Il peut reposer sur diverses technologies de génération d'interface utilisateurs ;
- Le **serveur d'application,** dont le rôle est de servir de plateforme d'exécution pour les applications de l'entreprise. Il peut reposer sur un serveur JEE (IBM WebSphere, Oracle Weblogoc, Tomcat, etc.), sur la paletforme Microsoft .Net, etc..
- Le **système de persistance**, dont le rôle est de stocker et de garantir l'intégrité des données métiers de l'entreprise. Il peut reposer sur une base de données relationnelle (Oracle, Microsoft SQL server, MySQL, etc) ou bien sur un serveur de fichiers, une base documentaire, etc.
- Le **serveur d'authentification/gestion de droits**, dont le rôle est de fournir des services de sécurité aux applications de l'entreprise. Il peut reposer sur un annuaire LDAP[8], un système de SSO[9], etc.
- Le **serveur d'intégration**, dont le rôle est de fournir une passerelle d'échange avec les autres applications de l'entreprise. Il peut reposer sur un middleware orienté messages (MOM), un ESB[10], etc.

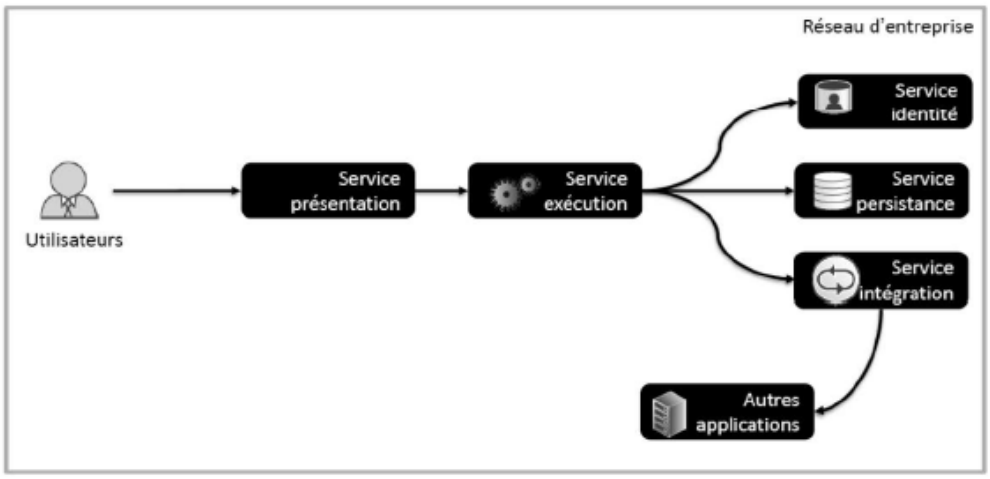


Figure 3 : Représentation schématisation d'une architecture N-Tiers

[8] Lighweight Directory Acess Protocol
[9] Single Sign On
[10] Entreprise Service Bus

7.2 Cloud Computing et Architectures de services

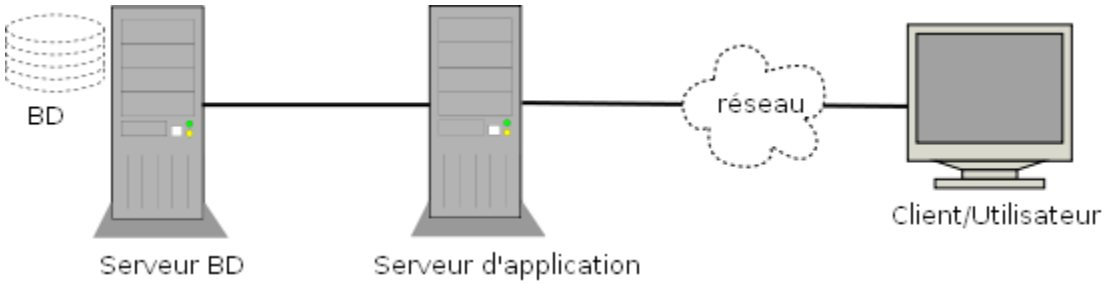
Les principes de l'architecture du Cloud Computing :

- N-tiers
- SOA
- Machine virtuelle
- Virtualisation des fichiers

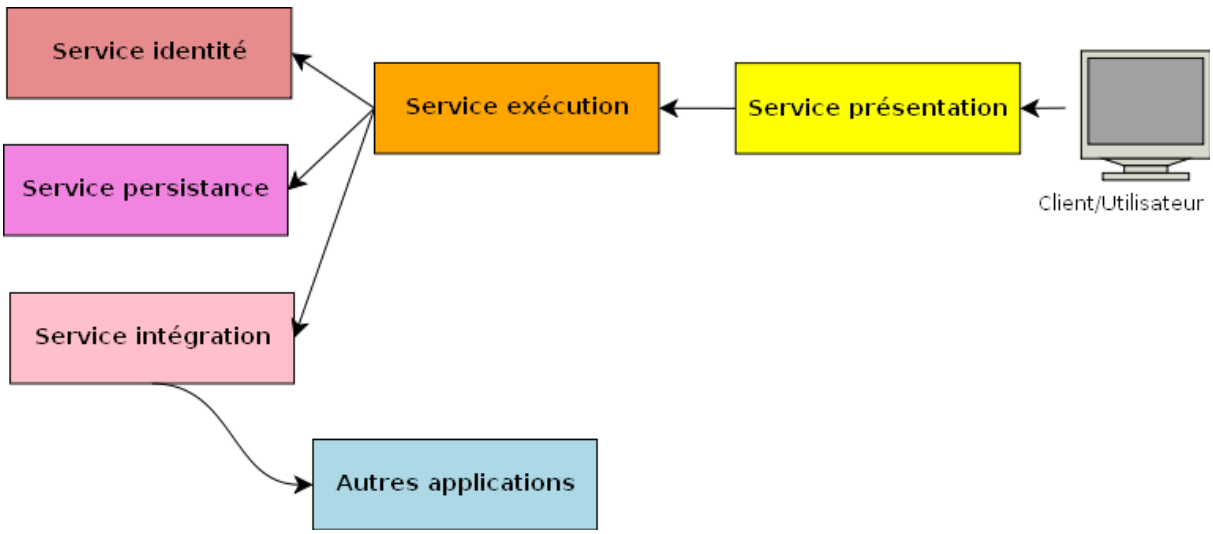
Architecture N-Tiers :

La philosophie des architectures du Cloud Computing repose sur l'architecture 3-tiers (N(Tiers)

Le principe de l'architecture 3-tiers :



L'architecture en N-tiers structuré en services (au lieu de serveurs) :

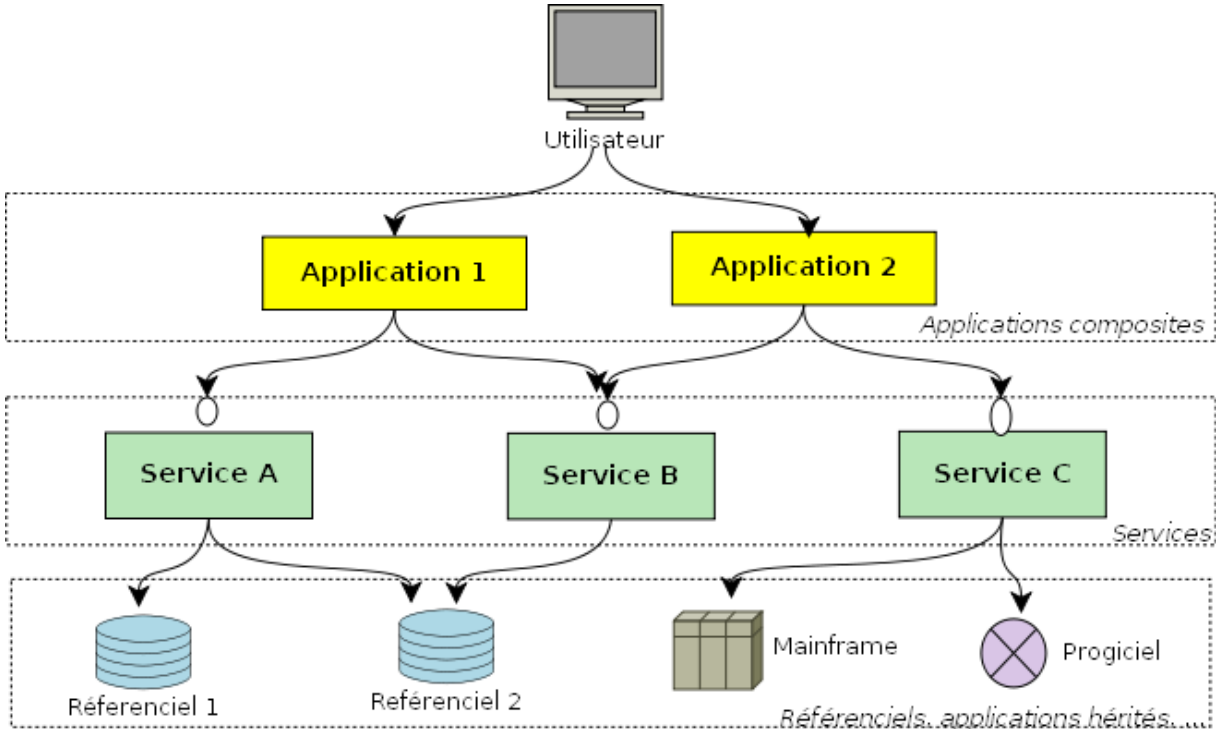


- **Serveur de présentation** : produit des écrans visibles par les utilisateurs (interfaces utilisateur)
- **Serveur d'application** : rôle de plate-forme d'exécution pour les applications de l'entreprise.

- **Système de persistance** : rôle de stockage des données métiers de l'entreprise. Il est basé sur un SGBD relationnel ou système de fichiers ou SGBD XML/ODD/ autres.
- **Serveur d'authentification/gestion d'identités** : assure les services de sécurité aux applications du SI (peut utiliser un annuaire LDAP, un système de SSO).
- **Serveur d'intégration** : fournit une passerelle d'échange avec les autres applications du SI.

Architecture SOA :

SOA (Service Oriented Architecture) : Architecture orientée services- les applications sont des assemblages des services métiers et des services génériques. Un service est une fonctionnalité orienté-métier.



Une architecture orienté service est une architecture logicielle s'appuyant sur un ensemble de services simples. Ils sont développés en s'inspirant des processus métier de l'entreprise. Un service est un composant fonctionnant de manière autonome et offrant des fonctionnalités métiers à d'autres applications ou d'autres services. Ces services représentent les fonctions basiques des fonctionnalités des entreprises. Ils dialoguent entre eux au travers de bus ou par Internet, on parle alors de *WebService* (WSOA). Les échanges peuvent se faire de manière synchrone ou asynchrone. L'entreprise s'enrichit de services mutualisables permettant de répondre rapidement et avec souplesse aux demandes du marché. En effet, ils correspondent à un processus métier mutualisable au niveau de l'entreprise. Cela permet les changements au niveau informatique des décisions stratégiques et tactiques de l'entreprise.
Il n'existe pas spécifications officielles pour l'architecture d'une SOA. Il peut-être décrit par la notion, la description, la publication et l'invocation de services. La notion de service est une fonction encapsulée dans un composant que l'on peut interroger à l'aide d'une requête composée d'un ou plusieurs paramètres et fournissant une ou plusieurs réponses. Idéalement chaque service doit être indépendant des autres afin de garantir sa réutilisabilité et son interopérabilité. La description de service est la manière de décrire les paramètres d'entrée du

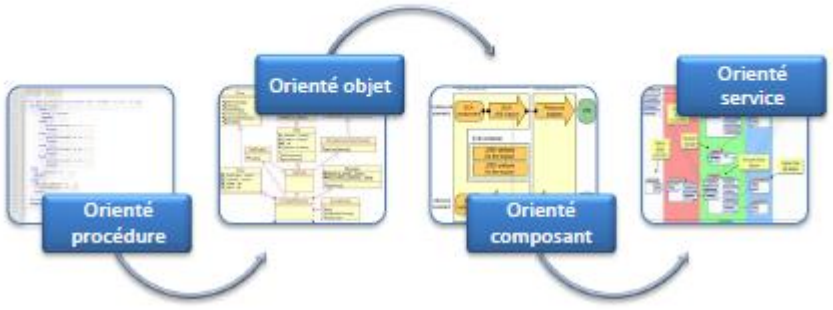
service, le format et le type des données retournées. Le principal format de description de services est WSDL (*Web Services Description Language*), normalisé par le W3C. Ensuite la publication consiste à publier dans un registre (en anglais *registry* ou *repository*) les services disponibles aux utilisateurs, tandis que la notion de découverte recouvre la possibilité de rechercher un service parmi ceux qui ont été publiés. Le principal standard utilisé est UDDI (*Universal Description Discovery and Integration*), normalisé par l'OASIS. Enfin l'invocation représente la connexion et l'interaction du client avec le service. Le principal protocole utilisé pour l'invocation de services est SOAP (*Simple Object Access Protocol*).

Les services au centre de ce schéma sont les unités atomiques d'une architecture orientée services. Ces services communiquent entre eux par le biais de messages. Ces communications peuvent être synchrones ou asynchrones. Le langage de programmation du service n'a aucune importante, ainsi que la plateforme d'exécution, matérielle et logicielle. Ces services sont limités par un ensemble de contrats, c'est-à-dire qu'un service doit respecter un ensemble de règle de fonctionnement.

Enfin dans les WSOA (*WebService Oriented Architecture*), les *WebServices* sont des services disponibles via Internet par le biais d'URL. Pour les WSOA, les données sont stockées sur le serveur. On peut se connecter aux *WebServices* à l'aide de terminaux mobiles. L'utilisateur dispose donc à partir d'un client léger la puissance de calcul du serveur. La connexion à distance est réalisé par protocole http, on a donc une indépendance de tout langage de programmation et plateformes. La communication entre les différents acteurs du système est réalisée par la technologie XML. Le schéma suivant montre le fonctionnement d'un WSOA entre différents terminaux et les *WebServices*.

Cycle de vie des services

Une des plus grandes fonctionnalités des SOA est la réutilisation des services pour plusieurs entreprises. En effet, en étudiant l'évolution des différentes architectures, on peut observer une réutilisation croissante. Ceci était impossible dans le modèle des composants.



En passant du procédural à l'orienté objet, on a la notion de réutilisation de classes d'objet. Celle-ci n'a cessé d'augmenter avec la vision orienté composant puis enfin service.

Virtual Machine :

Le concept de virtualisation désigne l'émulation complète, en isolation et en temps réel des environnements différents (systèmes d'exploitation) sur un même serveur. On obtient de cette manière deux ou plusieurs machines virtuelles qui fonctionnent sur un même serveur physique.

Exemples d'émulateurs : CYGWIN, machine virtuelle JAVA.

Virtual du stockage :

La virtualisation du stockage repose sur le principe qu'un fichier sera gardé quelque part dans le réseau et pourra être manipulé à tout moment même via de protocoles standard.

Systèmes de fichiers distribués :

- Google File System (GFS)
- Hadoop Distributed File System (HDFS)

Systèmes de Fichier de cluster :

- VMware vStorage (VMFS)
- XenServer Storage Pool

Chapitre 8 : Les architectures des services Web

8.1 Les fondations de l'architecture des services Web

L'architecture des services Web repose sur un mécanisme de transport d'une demande de service entre un client et un serveur, tous deux connectés au réseau.

Dans cette architecture, le *client* peut être soit un navigateur Web (la demande de service résulte alors directement d'une intervention humaine), soit une application (la demande de service est alors automatisée). Le rôle du *serveur*, quant à lui, est joué par une application qui s'exécute sur un moteur de scripts interfacé à un serveur HTTP (PHP/ Apache, par exemple) ou sur un serveur d'applications J2EE ou .NET. Le point crucial est ici l'étanchéité entre les clients et les serveurs qui ignorent tout de l'implémentation, dite privée, des opérations respectives de leurs correspondants et ne se connaissent et se reconnaissent qu'au travers de descriptions publiques, appelées *services Web*. À partir de ces services Web, on construit soit de nouveaux services Web dérivés des précédents, soit des applications Web, c'est-à-dire des assemblages de services Web correspondant à une solution fonctionnelle ou métier de l'entreprise. Le service Web est un composant logiciel dans la constitution d'applications métier s'appuyant sur l'infrastructure de communication offerte par le Web. Le mécanisme de communication permettant cette circulation de requêtes et de résultats autorise évidemment la mise en relation de plusieurs clients et de plusieurs serveurs et, bien souvent, d'ailleurs, d'applications jouant parfois le rôle de client et parfois celui de serveur.

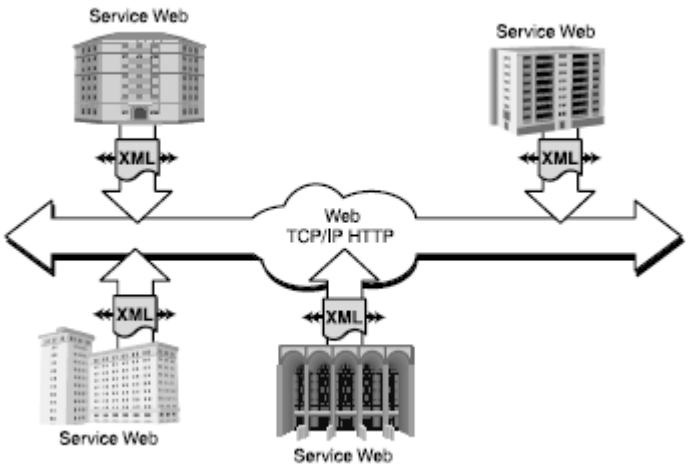


Figure 4: contexte d'un service web

Ce « bus de requêtes » est fondé sur TCP/IP et sur HTTP, ce qui permet son utilisation tant sur Internet qu'en vue de l'intégration d'applications au sein du réseau interne de l'entreprise ou de la « publication » d'applications préexistantes sur le Web à destination des entreprises partenaires.
Le service Web est l'interface publique de l'application dans le réseau intra ou interentreprise.

Comme HTTP ne transmet que du texte – des pages Web au format HTML, par exemple –, tous les échanges entre services Web (requêtes et résultats de requêtes) circulent également au format texte, sous forme de documents codés en XML.

8.2 Les éléments du service

Une application logicielle qui exerce une activité dont les résultats sont directement ou indirectement exploitables par d'autres applications, éventuellement réparties sur un réseau, joue le rôle de *prestataire de services*. L'ensemble des résultats exploitables de l'activité est appelé *prestation de services*, et les applications qui en bénéficient jouent le rôle de *client du service*. Les termes « prestataire » et « client » correspondent à des rôles interprétés par les applications dans la relation de service. Une application peut être en même temps prestataire de plusieurs services distincts et cliente de différents services.
Une prestation de services réside dans l'ensemble des résultats de l'activité de l'application prestataire, qui peuvent être classés en trois groupes (voir figure 5) :

- *Informations* : l'application prestataire effectue pour le compte du client des traitements dont les résultats sont communiqués au client. Ce groupe comprend les applications à haute intensité de calcul (exemple : la mise en œuvre de modèles d'analyse numérique) aussi bien que les applications qui effectuent des recherches et des agrégations de données stockées sur des bases.

- *États* : l'application prestataire gère les états et les changements d'état, représentés par des ensembles de données (exemple : une base de données de gestion). Les états peuvent être volatiles, persistants (s'appuyant sur des données stockées sur mémoire secondaire) et durables (non seulement persistants, mais aussi capables de survivre à des défaillances de l'application prestataire et de son infrastructure, y compris de sa mémoire secondaire). Un changement d'état est en principe toujours réversible, mais il faut que l'application soit conçue et mise en œuvre à cet effet.

- *Effets de bord* : l'application prestataire effectue des interactions avec l'environnement, c'est-à-dire avec un ensemble de dispositifs qui permettent l'entrée et la sortie de données du système (exemple : l'impression d'une facture). Les effets de bords sont, à l'inverse des changements d'état, irréversibles par définition.

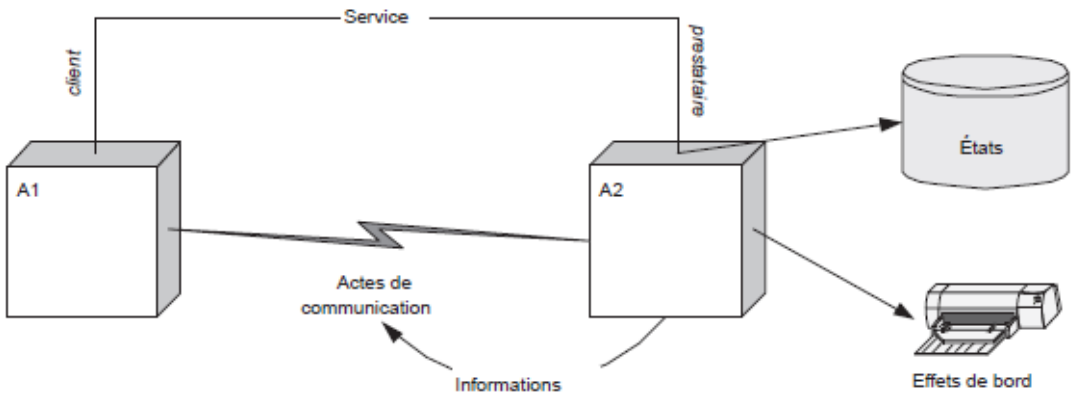


Figure 5: Les éléments d'une prestation de service

8.3 Les rôles de client et de prestataire

Une difficulté importante que l'on rencontre dans la maîtrise du modèle de l'architecture orientée services est la compréhension de la nature des *rôles* que sont ceux de client et de prestataire de services.
En effet, ce modèle s'applique, au-delà des architectures client/serveur, aux architectures réparties peer-to-peer, c'est-à-dire à des architectures réparties dans lesquelles il n'y a pas a priori de limitation de rôles pour les applications constituantes.
Une application qui fait partie d'une architecture orientée services peut être impliquée dans plusieurs relations de services et interpréter en même temps plusieurs rôles de client et plusieurs rôles de prestataire.
Une architecture orientée services est donc une *fédération de services* (voir figure 6), à savoir une architecture d'applications réparties qui participent à un réseau d'*échange de services*.
Il est bien évident que l'échange de services peut être circulaire : dans l'exemple de la figure 7, l'application A1 joue en même temps le rôle de client du service SA (exemple : un service d'achat en ligne) et le rôle de prestataire du service SB (exemple : un service de suivi de factures des achats effectués via le service SA) vis-à-vis de l'application A2, qui, à son tour, joue les rôles symétriques.
Nous appellerons par la suite une application qui a la capacité d'interpréter au moins le rôle de prestataire ou de client d'au moins une relation de service, une *application orientée services*.

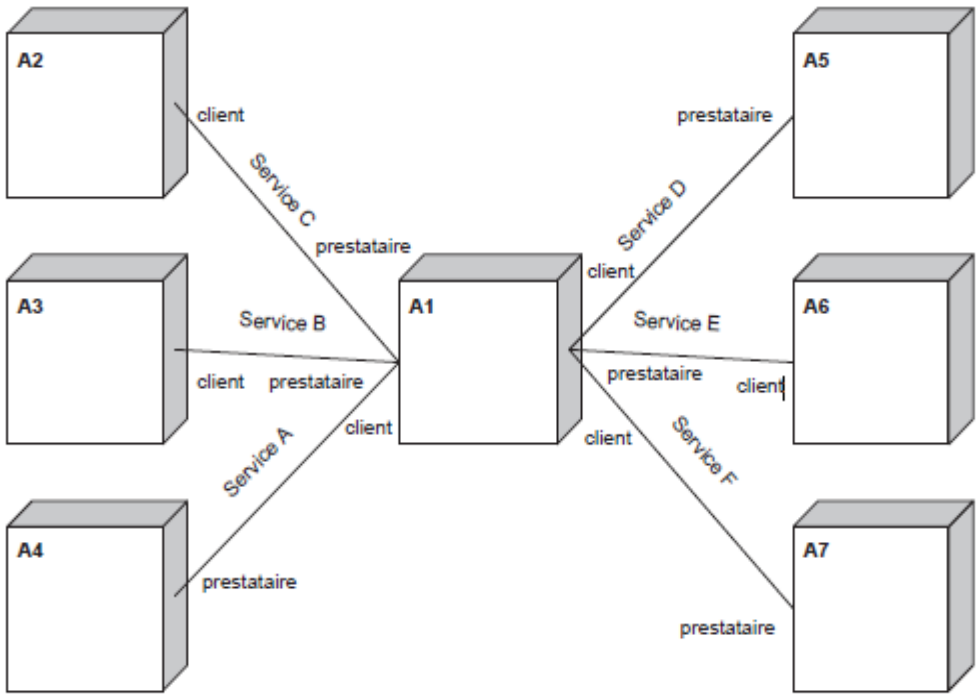


Figure 6: Une fédération de services

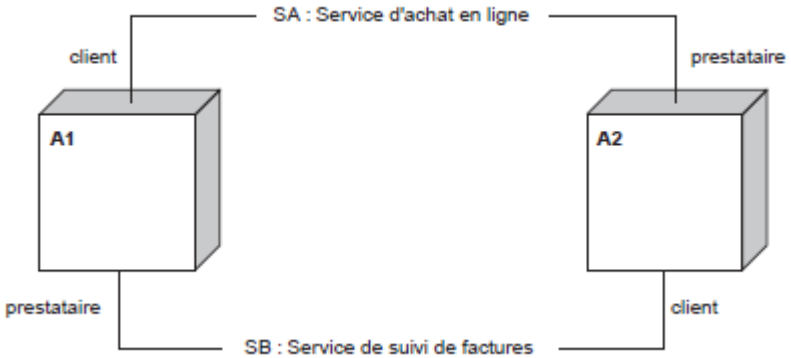


Figure 7:Echange circulaire de service

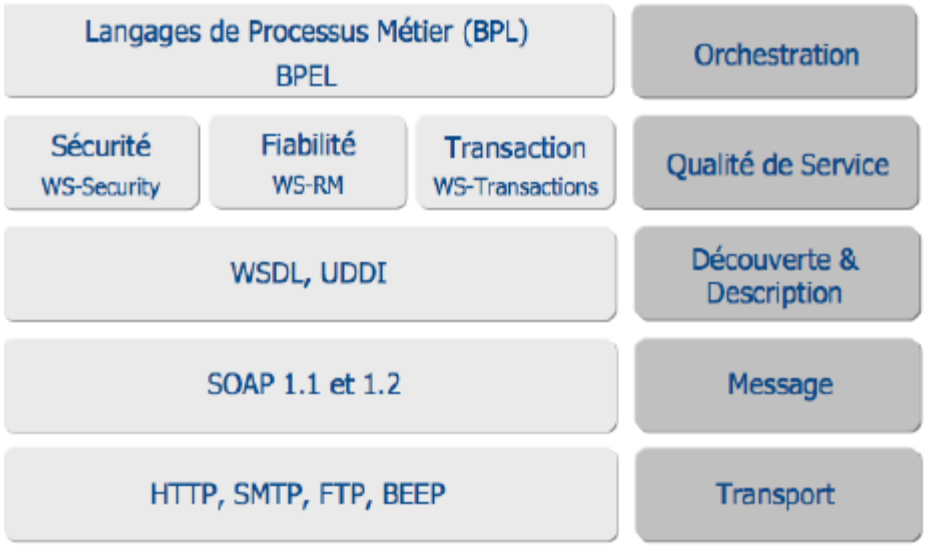
8.4 Le contrat de service

Le concept de service qui est véhiculé par le modèle de l'architecture orientée services se veut indépendant de la mise en œuvre des applications constituantes. Cette indépendance n'existe que s'il est possible de décrire (donc définir et découvrir) la relation de service indépendamment des implémentations des applications.
La description d'une relation de service est formalisée par un *contrat de service*. Ce contrat décrit les engagements réciproques du prestataire et du client du service. Toute application pouvant satisfaire les engagements du prestataire (et, réciproquement toute application pouvant satisfaire les engagements du client) peut interpréter le rôle de prestataire (et, réciproquement, le rôle du client).
Lorsqu'un contrat régente une relation de service, la réalisation de la prestation de services réside dans l'exécution du contrat.

8.5 Les standards des services Web

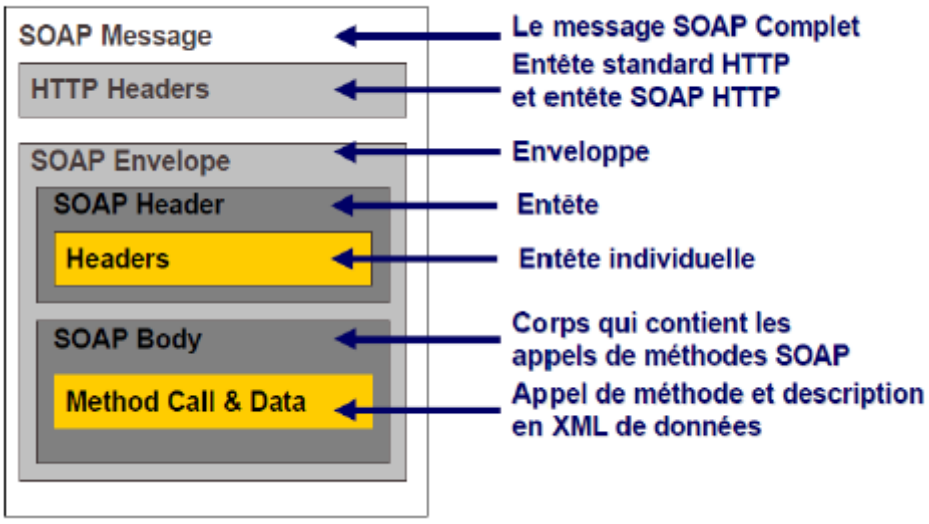
On comprend :

- http
- SOAP
- WSDL
- UDDI
- BPEL
- WS-Security



- **http** : HyperText Transfert Protocol) Protocole de communication sur Internet.
  - Disponible sur toutes les plateformes.
  - Simple
  - Sans connexion (peu de paquets sont nécessaires pour l'échange d'informations)
  - Niveau de sécurité simple et effectif
  - Le seul protocole utilisable à travers des pare-feu
- **SOAP** : Simple Object Access Protocol)
  - Protocole d'échange d'informations XML entre client et serveur http
  - Protocole minimal pour l'appel des méthodes sur des serveurs, services, composants, objets, et qui n'impose pas l'utilisation de :
    - API ou Runtime (API)
    - Serveur Web

-structure d'un message SOAP transporté par http



- WSDL : Web Service Description Language
    - Standard W3C basé sur XML pour la description d'interfaces des services Web
    - Une interface qui cache le détail de l'implémentation du service permettant une utilisation indépendante.
    - Le fichier WSDL est au format XML.
- UDDI : Universal Description Discovery and Integration
    - Annuaire mondial de services d'entreprises décrits en WSL et accessibles via des requêtes[11] SOAP.

[11] Un langage de **requête** est un langage **informatique** utilisé pour accéder aux données d'une base de données ou d'autres systèmes d'information. Il permet d'obtenir les données vérifiant certaines conditions (on parle de critères de sélection), comme toutes les personnes qui habitent une ville donnée.

Chapitre 9 : Le langage XML

9.1 Présentation

Le langage XML (eXtended Markup Language) est un format général de documents orienté texte. Il s'est imposé comme un standard incontournable de l'informatique. Il est aussi bien utilisé pour le stockage de documents que pour la transmission de données entre applications. Sa simplicité, sa flexibilité et ses possibilités d'extension ont permis de l'adapter à de multiples domaines allant des données géographiques au dessin vectoriel en passant par les échanges commerciaux. De nombreuses technologies se sont développées autour de XML et enrichissent ainsi son environnement.
Le langage XML dérive de SGML (Standard Generalized Markup Language) et de HTML (HyperText Markup Language). Comme ces derniers, il s'agit d'un langage orienté texte et formé de *balises* qui permettent d'organiser les données de manière structurée.

9.2 Pourquoi ce format

L'objectif est de connaître les principaux formats d'échanges de données entre logiciels : enregistrement de fichiers, communication entre serveur et clients, etc.

Les formats XML, JSON, YAML sont intéressants car ce sont des textes lisibles et structurés. Ils sont également accompagnés d'outils de vérification, transformation et interrogation. Ils sont directement liés à des structures de données dans la plupart des langages de programmation : un objet Java par exemple peut être lu ou écrit dans un fichier JSON ou XML.

Il faut noter que JavaScript semble prendre de plus en plus d'importance et il est fortement lié au format JSON.

9.3 Applications de XML

Le format XML est au cœur de nombreux processus actuels :

- format d'enregistrement de nombreuses applications,
- échange de données entre serveurs et clients,
- outils et langages de programmation,
- bases de données XML natives.

Il faut comprendre que XML définit la syntaxe des documents, mais les applications définissent les balises, leur signification et ce qu'elles doivent contenir. Des API existent dans tous les langages pour lire et écrire des documents XML.

9.4 Historique

9.4.1 Origine de XML

XML est un cousin de HTML. Tous deux sont les successeurs de SGML, lui-même issu de GML. Ce dernier a été conçu par IBM pour dissocier l'apparence des documents textes de leur contenu, en particulier des documentations techniques. Il est facile de changer l'apparence (mise en page, polices, couleurs. . .) d'un tel document sans en réécrire une seule ligne. D'autres langages, comme

LATEX[12] sont similaires : on écrit le texte sans se soucier de la mise en page. Cela permet également de le traduire dans d'autres langues.
Inversement, les logiciels *wysiwyg* comme Word mélangent mise en page et contenu. Même avec des styles, il reste difficile de remanier l'apparence d'un document sans devoir le réécrire partiellement.
XML permet de représenter beaucoup plus que des textes

9.4.2 Chronologie

- **1969 : GML (IBM)** GML est un langage d'écriture de documents techniques, défini par IBM, destiné à être traité par un logiciel propriétaire de mise en page appelé SCRIPT/VS.
- **1990 : SGML et HTML** SGML est une norme libre très complexe dont HTML est un sous ensemble très spécialisé. L'une des raisons qui ont conduit à les définir est d'améliorer la pérennité des documentations.
- **1998 : XML** c'est une généralisation de SGML permettant de construire toutes sortes de documents.

9.5 Langages apparentés

Un des atouts indéniables de XML est le nombre de technologies et de langages qui se sont développés autour de XML. Ceux-ci enrichissent les outils pour la manipulation des documents XML. La liste ci-dessous énumère les principaux langages qui font partie de l'environnement XML.

- XLink [ ] et XPointer [ ] (liens entre documents)

XML contient déjà un mécanisme pour matérialiser des liens entre des éléments d'un document. XLink et XPointer permettent d'établir des liens entre documents et plus particulièrement entre un élément d'un document et un fragment d'un autre document. Ils généralisent les liens hypertextes des documents HTML en autorisant des liens entre plusieurs documents.

- XPath [ ] (langage de sélection)

XPath est un langage d'expressions permettant de sélectionner des éléments dans un document XML. Il est la pierre angulaire du langage XSLT pour la transformation de documents.

- XQuery [ ] (langage de requête)

XQuery est un langage permettant d'extraire des informations à partir d'un ou plusieurs documents XML et de synthétiser de nouvelles informations à partir de celles extraites. Il s'apparente à un langage d'interrogation de bases de données et joue le rôle de SQL pour les documents XML.

- Schémas XML [ ] (modèles de documents)

Les schémas XML remplacent les DTD héritées de SGML pour décrire des modèles de documents. Ils sont beaucoup plus souples et beaucoup plus puissants que les DTD.

- XSLT [ ] (transformation de documents)

XSLT est un langage permettant d'exprimer facilement des transformations complexes entre documents XML.
Il s'appuie sur la structuration forte des documents XML vus comme des arbres. Chaque transformation est décrite par des règles pour chacun des éléments du document.

[12] Latex est un langage pour créer des rapports, des articles scientifiques et présentation en format pdf de haut qualité. Latex est basé par des balises.

9.6 Dialectes

De très nombreux dialectes ont été définis pour appliquer XML à des domaines très variés. Le grand avantage est que ces différents dialectes partagent la même syntaxe de base et que tous les outils XML peuvent être utilisés pour spécifier et manipuler ces documents. Il n'y a nul besoin de développer des outils spécifiques à ces différents dialectes. La liste ci-dessous énumère quelques uns de ces dialectes.

- RSS [ ] (Really Simple Syndication)

Abonnement à des flux de données

- XUL [ ] (XML-based User interface Language)

Langage de description d'interfaces graphiques développé par le projet Mozilla.

- SVG [ ] (Scalable Vector Graphics)

Description de dessins vectoriels

- SMIL [ ] (Synchronized Multimedia Integration Language)

Description de contenus multimédia

- MathML [ ] (Mathematical Markup Language)

Description de formules mathématiques

- WSDL [ ] (Web Services Description Language)

Description de services WEB

OpenStreetMap [ ]

Cartes libres

- XML Signature [ ]

Format pour les signatures électroniques

- SAML [ ] (Security Assertion Markup Language)

Langage d'échange d'authentifications et d'autorisations

- UBL [ ] (Universal Business Language)

Bibliothèque de documents standards pour les échanges commerciaux

- OpenDocument [ ]

Format de document pour les applications bureautiques développé au départ pour OpenOffice mais aussi utilisé par d'autres logiciels libres comme Calligra.

- DocBook [ ]

Format de documentation technique

De nombreux projets informatiques, comme Ant ou Android utilisent XML pour le stockage de données et en particulier pour les fichiers de configuration.

9.7 DocBook

DocBook est un exemple typique d'utilisation de XML. Il s'agit d'un format pour écrire des documents techniques.
Il est particulièrement adapté à la rédaction de documentations de logiciels. Il est d'ailleurs utilisé par de nombreux projets de logiciels libres, éventuellement de grande ampleur, comme le projet KDE. Cet ouvrage a été rédigé en utilisant DocBook. L'intégralité du texte est répartie en plusieurs fichiers XML. Afin d'obtenir une mise en page de qualité, les documents XML sont convertis, avec le langage XSLT, en un document LaTeX qui peut alors produire un document PDF.
DocBook était au départ basé sur SGML mais il s'appuie maintenant sur XML dont il est un des dialectes. Il contient de nombreuses balises permettant de décrire et de structurer le contenu de façon très précise. Il existe ensuite différents outils permettant de traduire un document DocBook, en une seule page HTML, en plusieurs pages HTML avec des liens ou encore en un document PDF.

DocBook met l'accent sur l'organisation et la structure du document. Son vocabulaire contient de très nombreuses balises permettant de transcrire très finement la sémantique de chaque fragment, à la manière de la seconde adresse donnée au début de cette introduction. Cet exemple est, en fait, très inspiré de DocBook. En revanche, DocBook ne permet pas de spécifier le rendu du document. Il n'est pas possible de donner, par exemple, la couleur ou la police de caractères à utiliser pour le texte. L'idée directrice est qu'un document DocBook doit permettre la production de plusieurs documents finaux à partir d'un même document original : document PDF, pages WEB. Comme les contraintes de ces différents média sont très diverses, il est impossible de pouvoir les spécifier dans le document.
Le choix de DocBook est de donner suffisamment d'indications sur le contenu aux applications qui réalisent les transformations pour obtenir un résultat de qualité. Les documents DocBook font souvent partie d'un ensemble de documentations, comme celle de KDE, dont la présentation est uniforme et donc déterminée de manière globale.

9.8 Syntaxe de XML

9.8.1 Premier exemple

Le langage XML est un format orienté texte. Un document XML est simplement une suite de caractères respectant quelques règles. Il peut être stocké dans un fichier et/ou manipulé par des logiciels en utilisant un codage des caractères. Ce codage précise comment traduire chaque caractère en une suite d'octets réellement stockés ou manipulés. Comme un document XML est seulement du texte, il peut être écrit comme l'exemple ci-dessous.
On commence par donner un premier exemple de document XML comme il peut être écrit dans un fichier `bibliography.xml`. Ce document représente une bibliographie de livres sur XML. Il a été tronqué ci-dessous pour réduire l'espace occupé. Ce document contient une liste de livres avec pour chaque livre, le titre, l'auteur, l'éditeur (publisher en anglais), l'année de parution, le numéro ISBN et éventuellement une URL.

```
<?xml version="1.0" encoding="iso-8859-1"?>                    (1)
<!-- Time-stamp: "bibliography.xml 3 Mar 2008 16:24:04" -->    (2)
<!DOCTYPE bibliography SYSTEM "bibliography.dtd" >             (3)
<bibliography>                                                 (4)
  <book key="Michard01" lang="fr">                             (5)
    <title>XML langage et applications</title>
    <author>Alain Michard</author>
    <year>2001</year>
    <publisher>Eyrolles</publisher>
    <isbn>2-212-09206-7</isbn>
    <url>http://www.editions-eyrolles/livres/michard/</url>
  </book>
  <book key="Zeldman03" lang="en">
   <title>Designing with web standards</title>
   <author>Jeffrey Zeldman</author>
   <year>2003</year>
   <publisher>New Riders</publisher>
   <isbn>0-7357-1201-8</isbn>
  </book>
   ...
</bibliography>                                                (6)
```

Descriptions :

1. Entête XML avec la version `1.0` et l'encodage `iso-8859-1` des caractères.
2. Commentaire délimité par les chaînes de caractères `<!--` et `-->`.
3. Déclaration de DTD externe dans le fichier `bibliography.dtd`.
4. Balise ouvrante de l'élément racine `bibliography`
5. Balise ouvrante de l'élément `book` avec deux attributs de noms `key` et `lang` et de valeurs `Michard01` et `fr`
6. Balise fermante de l'élément racine `bibliography`

9.8.2 Caractères

Un document XML est une suite de caractères. Les caractères qui peuvent être utilisés sont ceux définis par la norme Unicode ISO 10646 [ ] aussi appelée UCS pour *Universal Character Set*. Cette norme recense tous les caractères des langues connues et tous les symboles utilisés dans les différentes disciplines. Elle nomme tous ces caractères et symboles et leur attribue un code sur 32 bits (4 octets) appelé simplement *code Unicode* ou *point de code* dans la terminologie Unicode. Dans la pratique, tous les points de code attribués à des caractères se situent dans l'intervalle de 0 à 0x10FFFF et ils utilisent donc au plus 21 bits. Cette longueur maximale ne sera pas dépassée avant longtemps car il reste encore de nombreux points de code non attribués dans cet intervalle pour des usages futurs. Unicode peut être vu comme un catalogue de tous les caractères disponibles. Un caractère dont le point de code est *n* est désigné par U+*n* où le nombre *n* est écrit en hexadécimal. L'écriture hexadécimale de *n* n'est pas préfixée du caractère 'x' car c'est implicite après les deux caractères 'U+'. Le caractère Euro '€' est, par exemple, désigné par U+20AC car son point de code est 8364 = 0x20AC. Le sous-ensemble des caractères Unicode dont les points de code tiennent sur 16 bits (2 octets), c'est-à-dire entre 0 et 65535 = 0xFFFF est appelé BMP pour *Basic Multilingual Plane*. Il couvre largement la très grande majorité des langues usuelles et les symboles les plus courants.

9.8.2.1 Caractères spéciaux

Les cinq caractères '<' U+2C, '>' U+2E, '&' U+26, ''' U+27 et '"' U+22 ont une signification particulière dans les documents XML. Les deux caractères '<' et '>' servent à délimiter les balises, ainsi que les commentaires et les instructions de traitement. Le caractère '&' marque le début des références aux entités générales. Pour introduire ces caractères dans le contenu du document, il faut utiliser des sections littérales ou les entités prédéfinies. Les caractères ''' et '"' servent également de délimiteurs, en particulier pour les valeurs des attributs. Dans ces cas, il faut encore avoir recours aux entités prédéfinies pour les introduire.

9.8.2.2 Caractères d'espacement

Le traitement XML des caractères d'espacement est simple dans un premier temps. Chaque caractère d'espacement est équivalent à un espace et plusieurs espaces consécutifs sont encore équivalents à un seul espace. C'est le comportement habituel des langages de programmation classiques. Dans un second temps, le traitement des caractères d'espacement par une application est commandé par l'attribut xml:space. Les caractères d'espacement sont l'espace ' ' U+20, la tabulation U+09 ('\t' en notation du langage C), le saut de ligne U+0A ('\n' en C) et le retour chariot U+0D ('\r' en C).

Les fins de lignes sont normalisées par l'analyseur lexical (*parser* en anglais). Ceci signifie que les différentes combinaisons de fin de ligne sont remplacées par un seul caractère U+0A avant d'être transmises à l'application.
Cette transformation garantit une indépendance vis à vis des différents systèmes d'exploitation. Les combinaisons remplacées par cette normalisation sont les suivantes.

- la suite des deux caractères U+0D U+0A
- la suite des deux caractères U+0D U+85
- le caractère U+85 appelé *Next Line*
- le caractère U+2028 appelé *Line Separator*
- le caractère U+0D non suivi de U+0A ou U+85

Les deux caractères U+85 et U+2028 ne peuvent être correctement décodés qu'après la déclaration de l'encodage des caractères par l'entête. Leur usage dans l'entête est donc déconseillé.

9.8.2.3 Jetons et noms XML

Les identificateurs sont utilisés en XML pour nommer différents objets comme les éléments, les attributs, les instructions de traitement. Ils servent aussi à identifier certains éléments par l'intermédiaire des attributs de type ID. XML distingue deux types d'identificateurs: les *jetons* et les *noms XML*. La seule différence est que les noms XML doivent commencer par certains caractères particuliers. Dans la terminologie XML, les jetons sont appelés NMTOKEN pour *name token*.
Les caractères autorisés dans les identificateurs sont tous les caractères alphanumériques, c'est-à-dire les lettres minuscules [a-z], majuscules [A-Z] et les chiffres [0-9] ainsi que le tiret '-' U+2D, le point '.' U+2E, les deux points ':' U+3A et le tiret souligné '_' U+5F. Un *jeton* est une suite quelconque de ces caractères. Un*nom XML* est un jeton qui, en outre, commence par une lettre [a-zA-Z], le caractère ':' ou le caractère '_'.

Les deux caractères '-' et '.' ainsi que les chiffres ne peuvent pas apparaître au début des noms. Il n'y a pas, a priori, de limite à la taille des identificateurs mais certains logiciels peuvent en imposer une dans la pratique.
Le caractère ':' est réservé à l'utilisation des espaces de noms [Chapitre 4]. De fait, il ne peut apparaître qu'une seule fois pour séparer un préfixe du nom local dans les noms des éléments et des attributs. Les espaces de noms amènent à distinguer les noms ayant un caractère ':', appelés *noms qualifiés* et les autres, appelés par opposition *noms non qualifiés*.
Les noms commençant par les trois lettres xml en minuscule ou majuscule, c'est-à-dire par une chaîne de [xX][mM][lL] sont réservés aux usages internes de XML. Ils ne peuvent pas être utilisés

librement dans les documents mais ils peuvent cependant apparaître pour des utilisations spécifiques prévues par la norme. Les noms commençant par xml: comme xml:base font partie de l'espace de noms XML.

Quelques exemples d'identificateurs sont donnés ci-dessous.

Noms XML valides :
`name`, `id-42`, `xsl:template`, `sec.dtd-3.1` et `_special_`
Jetons qui ne sont pas des noms :
`-name`, `42`, `42-id` et `.sect.`
Noms réservés :
`xml:id` et `xml-stylesheet`

9.8.2.4 Codage

Chaque caractère possède un point de code sur 32 bits mais un document ne contient pas directement ces points de code des caractères. Ce codage serait inefficace puisque chaque caractère occuperait 4 octets. Chaque document utilise un codage pour écrire les points de code des caractères. Il existe différents codages dont le codage par défaut UTF-8. Certains codages permettent d'écrire tous les points de code alors que d'autres permettent seulement d'écrire un sous-ensemble comme le BMP. Le codage utilisé par un document est indiqué dans l'entête de celui-ci. Les principaux codages utilisés par les documents XML sont décrits ci-dessous.

- US-ASCII

Ce codage permet uniquement de coder les points de code de 0 à 0x7F des caractères ASCII.

- UCS-4 ou UTF-32 [ ]

Chaque caractère est codé directement par son point de code sur quatre octets. Ce codage permet donc de coder tous les caractères Unicode.

- UCS-2 [ ]

Chaque caractère est codé par son point de code sur deux octets. Ce codage permet donc uniquement de coder les caractères du BMP.

- UTF-16 [ ]

Ce codage coïncide essentiellement avec UCS-2 à l'exception de la plage de 2048 positions de 0xD800 à 0xDFFF qui permet de coder des caractères en dehors du BMP dont le point de code utilise au plus 20 bits.
L'exclusion de cette plage ne pose aucun problème car elle ne contient aucun point de code attribué à un caractère. Un point de code ayant entre 17 et 20 bits est scindé en deux blocs de 10 bits répartis sur une paire de mots de 16 bits. Le premier bloc de 10 bits est préfixé des 6 bits 110110 pour former un premier mot de 16 bits et le second bloc de 10 bits est préfixé des 6 bits 110111 pour former un second mot de 16 bits.

Représentation UTF-16	**Signification**
`xxxxxxxx xxxxxxxx`	2 octets codant 16 bits
`110110``zz zzxxxxxx`	4 octets codant 20 bits
`110111``xx xxxxxxxx`	yyyy xxxxxxxx xxxxxxxx où zzzz = yyyy-1

Le symbole de l'Euro '€' U+20AC, est, par exemple, codé par les deux octets x20 xAC = 00100000 10101100 puisqu'il fait partie du BMP. Le symbole de la croche '##' U+1D160 est codé par les 4 octets xD8 x34 xDD x60 = **110110**00 00110100 **110111**01 01100000.

- UTF-8 [ ]

Ce codage est le codage par défaut de XML. Chaque caractère est codé sur un nombre variable de 1 à 4 octets. Les caractères de l'ASCII sont codés sur un seul octet dont le bit de poids fort est 0. Les caractères en dehors de l'ASCII utilisent au moins deux octets. Le premier octet commence par autant de 1 que d'octets dans la séquence suivis par un 0. Les autres octets commencent par 10. Ce codage peut uniquement coder des points de code ayant au maximum 21 bits mais tous les points de code attribués ne dépassent pas cette longueur.

Représentation UTF-8	**Signification**
0xxxxxxx	1 octet codant 7 bits
110xxxxx **10**xxxxxx	2 octets codant 8 à 11 bits
1110xxxx **10**xxxxxx **10**xxxxxx	3 octets codant 12 à 16 bits
11110xxx **10**xxxxxx **10**xxxxxx **10**xxxxxx	4 octets codant 17 à 21 bits

Le symbole de l'Euro '€' U+20AC, est, par exemple, codé par les trois octets xE2 x82 xAC = **1110**0010 **10**000010 **10**101100. Le symbole de la croche '𝅘𝅥𝅯' U+1D160 est, quant à lui, codé par les 4 octets xF0 x9D x85 xA0 = **11110**000 **10**011101 **10**000101 **10**100000. Ce codage a l'avantage d'être relativement efficace pour les langues européennes qui comportent beaucoup de caractères ASCII. Il est, en revanche, peu adapté aux langues asiatiques dont les caractères nécessitent 3 octets alors que 2 octets sont suffisants avec UTF-16.

- ISO-8859-1 (appelé Latin-1) [ ]

Chaque caractère est codé sur un seul octet. Ce codage coïncide avec l'ASCII pour les points de code de 0 à 0x7F. Les codes de 0x80 à 0xFF sont utilisés pour d'autres caractères (caractères accentués, cédilles, …) des langues d'Europe de l'ouest.

- ISO-8859-15 (appelé Latin-9 ou Latin-0) [ ]

Ce codage est une mise à jour du codage ISO-8859-1 dont il diffère uniquement en 8 positions. Les caractères '€', 'Š', 'š', 'Ž', 'ž', 'OE', 'oe' et 'Ÿ' remplacent les caractères '¤' U+A4, '¦' U+A6, '¨' U+A8, '´' U+B4, '¸' U+B8, '¼' U+BC, '½' U+BD et '¾' U+BE moins utiles pour l'écriture des langues européennes. Le symbole de l'Euro '€' U+20AC, est, par exemple, codé par l'unique octet xA4 = 10100100.
Le tableau suivant donne la suite d'octets pour le mot Hétérogène pour quelques codages classiques. Comme les points de code de x00 à xFF d'Unicode coïncident avec le codage des caractères de ISO-8859-1, le codage en UTF-16 est obtenu en insérant un octet nul 00 avant chaque octet du codage en ISO-8859-1. Le codage en UTF-32 est obtenu en insérant dans le codage en UTF-16 deux octets nuls avant chaque paire d'octets.

Tableau comparatif des différents codages :

Codage	**Séquence d'octets en hexadécimal pour `Hétérogène`**
UTF-8	48 C3 A9 74 C3 A9 72 6F 67 C3 A8 6E 65
ISO-8859-1	48 E9 74 E9 72 6F 67 E8 6E 65
UTF-16	00 48 00 E9 00 74 00 E9 00 72 … 00 E8 00 6E 00 65
UTF-32	00 00 00 48 00 00 00 E9 00 00 00 74 … 00 00 00 65

9.8.3 URI, URL et URN

XML et toutes les technologies autour de XML font un grand usage des URI et plus particulièrement des URL.
Ceux-ci sont, par exemple, employés pour référencer des documents externes comme des DTD ou pour identifier des espaces de noms. Les URL sont bien connues car elles sont utilisées au quotidien pour naviguer sur le WEB. La terminologie XML distingue également les URI et les URN. Les significations exactes de ces trois termes dans la terminologie XML sont les suivantes.

- URI

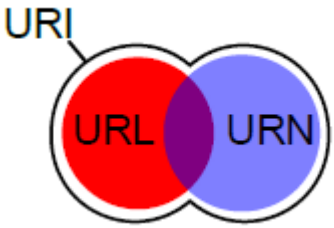
Uniform Resource Identifier

- URL

Uniform Resource Locator

- URN

Uniform Resource Name



9.8.3.1 Résolution d'URI

Un URI appelé *URI de base* est souvent attaché à un document ou à un fragment d'un document XML. Cet URI est généralement une URL. Il sert à résoudre les URL contenues dans le fragment de document, qu'elles soient relatives ou absolues. Cette *résolution* consiste à combiner l'URL de base avec ces URL pour obtenir des URL absolues qui permettent de désigner des documents externes.

Pour comprendre comment une URL de base se combine avec une URL, il faut d'abord comprendre la structure d'une URL. La description donnée ci-dessous se limite aux aspects indispensables pour appréhender la résolution des URL. Chaque URL se décompose en trois parties.

- **Protocole d'accès**

Une URL commence obligatoirement par le nom d'un protocole d'accès suivi du caractère ':'. Les principaux protocoles sont http, https, ftp et file.

- **Adresse Internet**

Le protocole est suivi d'une adresse Internet qui commence par les deux caractères '//'. Cette adresse est absente dans le cas du protocole file.

- **Chemin d'accès**

L'URL se termine par un chemin d'accès dans l'arborescence des fichiers. Ce chemin se décompose lui-même en le nom du répertoire et le nom du fichier. Ce dernier est formé de tous les caractères après le dernier caractère '/'.

Nous allons décrire cette url : http://www.dimitripianeta.fr/cours/mathematiques.html

Le protocole est http : . qui est le protocole http, adresse Internet est www.dimitripianeta.fr et le répertoire est cours et le fichier est mathematiques.html. L'ensemble du répertoire «cours » et le fichier « mathematiques.html » correspond au chemin d'accès.

9.9 Structure d'un document XML

9.9.1 Arborescence d'éléments

Un document XML est composé de plusieurs parties :

• Entête de document précisant la version et l'encodage,

• Des règles optionnelles permettant de vérifier si le document est valide (c'est l'objet des deux prochains cours),

• Un arbre d'*éléments* basé sur un élément appelé *racine*

– Un *élément* possède un *nom*, des *attributs* et un *contenu*

– Le contenu d'un élément est :

* du texte
* d'autres éléments (les éléments enfants).

– Un élément est marqué par une *balise ouvrante* et une *balise fermante*.

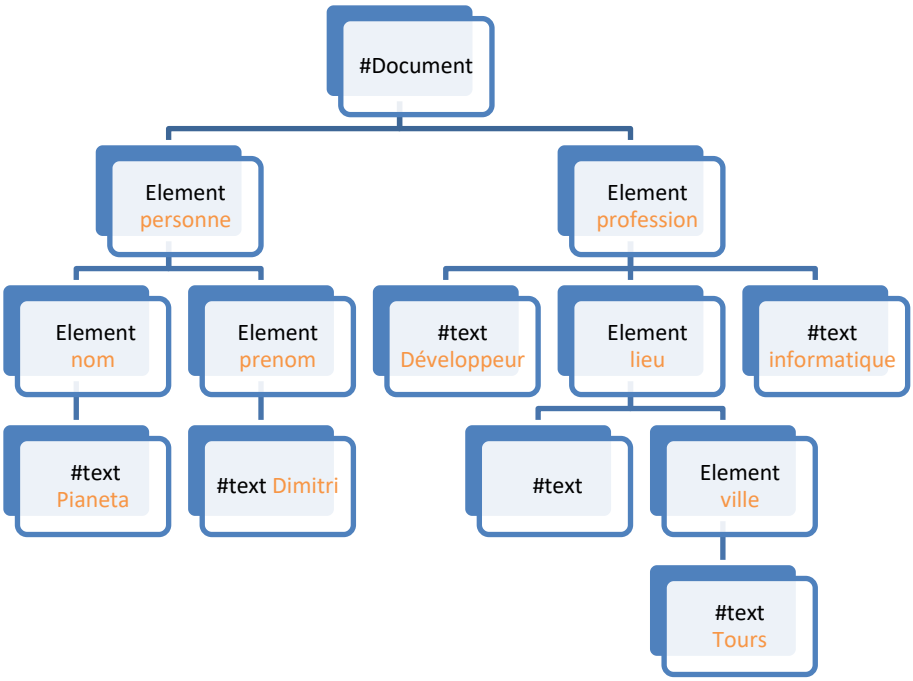
* une balise ouvrante est notée <nom attributs...>
* une balise fermante est notée </nom>

9.9.2 Exemple complet

```
<?xml version="1.0" encoding="utf-8"?>
<personne>
        <nom>Pianeta</nom>
        <prenom>Dimitri</prenom>
        <profession>
                Developpeur
            <lieu>

                <ville>Tours</ville>
            </lieu>
          informatique
      </profession>
</personne>
```

9.9.3 Représentation graphique



Explication :

Le document XML représente un arbre composé de plusieurs types de noeuds :

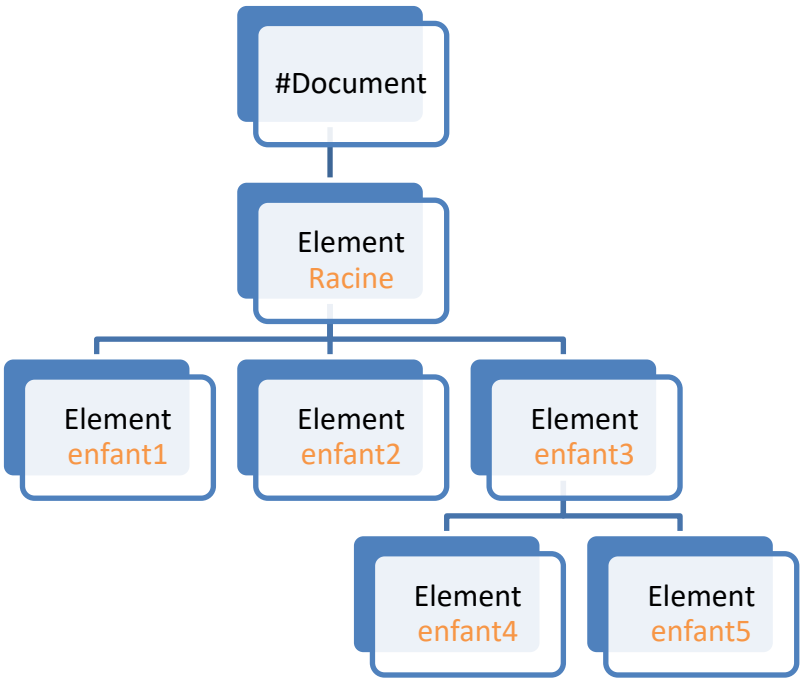
- **Nœuds éléments** ils sont associés aux balises <element>. Ce sont des nœuds qui peuvent avoir des enfants en dessous.
- **Nœuds #text** ils représentent le texte situé entre deux balises. Les nœuds texte sont des feuilles dans l'arbre.
-

Notez que différents textes peuvent être entrelacés avec des éléments. Voir le cas de <lieu> dans le contenu de <profession>. Il est possible de le faire, mais ce n'est pas forcément souhaitable.
Les autres types de nœuds seront présentés au fur et à mesure.

Vocabulaire :

Voici comment on désigne les différents noeuds les uns par rapport aux autres :

- <racine> est le noeud **parent** du noeud **enfant** (*child*) <enfant3>, lui-même parent de <enfant4> et <enfant5>,
- <racine>, <enfant3> sont des noeuds **ancêtres** (*ancestors*) de <enfant4> et <enfant5>,
- <enfant4> et <enfant5> sont des **descendants** (*descendants*) de <racine> et <enfant3>,
- <enfant1> est un noeud **frère** (*sibling*) de <enfant2> et réciproquement.



9.9.4 Détails du format XML

9.9.4.1 Prologue

La première ligne d'un document XML est appelée *prologue XML*. Elle spécifie la version de la norme XML utilisée (1.0 ou 1.1 qui sont très similaires[13]) ainsi que l'encodage :

```
<?xml version="1.0" encoding="utf-8"?>
<?xml version="1.0" encoding="iso-8859-15"?>
```

9.9.4.2 Norme Unicode

La norme Unicode définit un ensemble abstrait de plus de 245000 caractères représentable sur un ordinateur, par exemple é, €, «, ©. . . Ces caractères doivent être codés sous forme d'octets. C'est là qu'il y a plusieurs normes :

- ASCII ne représente que les 128 premiers caractères Unicode.
- ISO 8859-1 ou Latin-1 représente 191 caractères de l'alphabet latin n°1 (ouest de l'Europe) à l'aide d'un seul octet. Les caractères € et oe n'en font pas partie, pourtant il y a æ.
- ISO 8859-15 est une norme mieux adaptée au français. Elle rajoute € et oe et supprime des caractères peu utiles comme ¤.
- UTF-8 représente les caractères à l'aide d'un nombre variable d'octets, de 1 à 4 selon le caractère.
  Par exemple : A est codé par 0x41, é par 0xC3,0xA9 et € par 0xE2,0x82,0xAC.

Pour en savoir, je vous conseils de revoir le paragraphe 9.8.2.4 Codage.

[13] La norme 1.1 est plus claire concernant certains nouveaux caractères unicode qui peuvent spécifier des retours à la ligne et autres caractères de contrôle.

9.9.4.3 Commentaire

On peut placer des commentaires à peu près partout dans un document XML. La syntaxe est identique à celle d'un fichier HTML. Un commentaire peut d'étendre sur plusieurs lignes. La seule contrainte est de ne pas pouvoir employer les caractères -- dans le commentaire, même s'ils ne sont pas suivis de >

```
<!-- voici un commentaire -->
ceci n'est pas un commentaire
<!--
        voici un autre commentaire
        et ça -- , c'est une erreur
-->
```

9.9.4.4 Attention aux -– dans les commentaires

En fait, en XML (comme en SGML), <!-- et --> ne sont pas des marques de commentaires comme /* et */ en C.

- <! marque le début d'une instruction de traitement destinée à l'analyseur
- > signale la fin de cette instruction de traitement
- -- inverse le mode « commentaire », entre « hors commentaire » et « dans un commentaire ».

Ainsi, un commentaire <!--commentaire--> est analysé ainsi :

1. <! début d'une instruction de traitement
2. -- passage en mode commentaire
3. commentaire : ignoré, on peut donc tout y mettre sauf --
4. -- sortie du mode commentaire, ne rien mettre après
5. > sortie de l'instruction de traitement

9.9.4.5 Options après le prologue

Après le prologue, on peut trouver plusieurs parties optionnelles délimitées par <?...?> ou <!...>. Ce sont des instructions de traitement, des directives pour l'analyseur XML.

Par exemple :

- un *Document Type Definitions* (DTD) qui permet de valider le contenu du document. C'est l'objet du prochain cours.
- une feuille de style,

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE personne SYSTEM "personne.dtd">
<?xml-stylesheet href="personne.css" type="text/css"?>
<personne>
...
</personne>
```

9.9.4.6 Eléments

Les éléments définissent la structure du document. Un élément est délimité par :

- une balise ouvrante <nom attributs...>
- une balise fermante </nom> obligatoire.

Le contenu de l'élément se trouve entre les deux balises. Ce sont des éléments enfants et/ou du texte.

```
<parent>
    texte1
    <enfant>texte2</enfant>
    texte3
</parent>
```

Dans le cas où l'élément n'a pas de contenu (*élément vide*), on peut regrouper ses deux balises en une seule <nom attributs.../>.

9.9.4.7 Choses interdites

Les règles d'imbrication XML interdisent différentes configurations qui sont plus ou moins tolérées en HTML :

- plusieurs racines dans le document,
- des éléments non terminés (NB: XML est sensible à la casse),
- des éléments qui se chevauchent.

```
<element1>
    <element2>
    </Element2>
    <element3>
    </element1>
</element3>
```

En XML, cela crée des erreurs « *document mal formé* ».

9.9.4.8 Choses permises

Parmi les choses permises, le même type d'élément peut se trouver plusieurs fois avec le même parent, avec des contenus identiques ou différents (à vous de définir si ça a du sens) :

```
<element1>
    <element2>texte1</element2>
    <element2>texte2</element2>
    <element2>texte1</element2>
</element1>
```

Il est possible d'interdire cela, cependant ce n'est pas relatif à la syntaxe XML mais à la validation du document par rapport à une spécification, et aussi ce qu'on veut faire avec le document.

9.9.4.9 Noms des éléments

Les noms des éléments peuvent employer de nombreux caractères Unicode (correspondant au codage déclaré dans le prologue) mais pas les signes de ponctuation.

- L'initiale doit être une lettre ou le signe _
- ensuite, on peut trouver des lettres, des chiffres ou - . _

Le caractère : permet de séparer le nom en deux parties : préfixe et *nom local*. Le tout s'appelle *nom qualifié*. Par exemple departement:ville est un nom qualifié préfixé par departement. Ce préfixe permet de définir un *espace de nommage*.

nom qualifié = préfixe:*nom local*

9.9.4.10 Espaces de nommage

Un espace de nommage (*namespace*) définit une famille de noms afin d'éviter les confusions entre des éléments qui auraient le même nom mais pas le même sens. Cela arrive quand le document XML modélise les informations de plusieurs domaines.
Voici un exemple dans le domaine de la vente de meubles. Le document modélise une table (avec 4 pieds) et aussi un tableau HTML pour afficher ses dimensions. On voit la confusion.

```
<meuble id="765">
    <table prix="74,99€">acajou</table>
    <table border="1">
        <tr><th>longueur</th><th>largeur</th></tr>
        <tr><td>120cm</td><td>80cm</td></tr>
    </table>
</meuble>
```

9.9.5 Evolution de la norme

Une autre raison pour employer des espaces de nommage est de permettre l'extension d'une norme qui risque d'évoluer par la suite. Par exemple la norme GPX qui définit des fichiers de géolocalisation des GPS. Elle définit par exemple la structure d'un point de trace <trkpt>. Il a un certain nombre l'attributs comme lon et lat pour les coordonnées géographiques, et de sous-élements tels que <ele> qui mémorisent l'altitude d'un point.
La société Garmin a rajouté de nouvelles informations pour les sportifs tels que <hr> pour mémoriser le rythme cardiaque. Or il y a le risque que dans l'avenir, le format GPX évolue et définisse de son côté un élément <hr> avec une autre signification.
Grâce à un espace de nommage spécifique, la société Garmin peut définir ses propres balises <garmin:hr> sans risquer de conflit avec la norme GPX générale.

9.9.5.1 Définition d'un espace de nommage

On doit choisir un URI, par exemple un URL, pour identifier l'espace de nommage. Cet URI n'a pas besoin de correspondre à un véritable contenu car il ne sera pas téléchargé.

Un URI est la généralisation d'un URL :
schéma:[//[user:passwd@]hôte[:port]][/]chemin[?requête][#fragment],
par exemple http://en.wikipedia.org/wiki/Uniform_Resource_Identifier#Syntax.

Les URN sont au format urn:NID:NSS, ex: urn:iutlan:xmlsem1
Ensuite on rajoute un attribut spécial à la racine du document :

```
<?xml version="1.0" encoding="utf-8"?>
<racine xmlns:PREFIXE="URI">
        <PREFIXE:balise>...</PREFIXE:balise>
</racine>
```

9.9.5.2 Exemple revu

Voici l'exemple précédent, avec deux *namespaces*, un URN pour les meubles et un URL pour HTML :

```
<?xml version="1.0" encoding="utf-8"?>
<meuble:meuble id="765"
      xmlns:meuble="urn:iutlan:meubles
      xmlns:html="http://www.w3.org">
  <meuble:table prix="74,99€">acajou</meuble:table>
 <html:table border="1">
    <html:tr><html:th>longueur</html:th>...</html:tr>
   <html:tr><html:td>120cm</html:td>...</html:tr>
  </html:table>
</meuble:meuble>
```

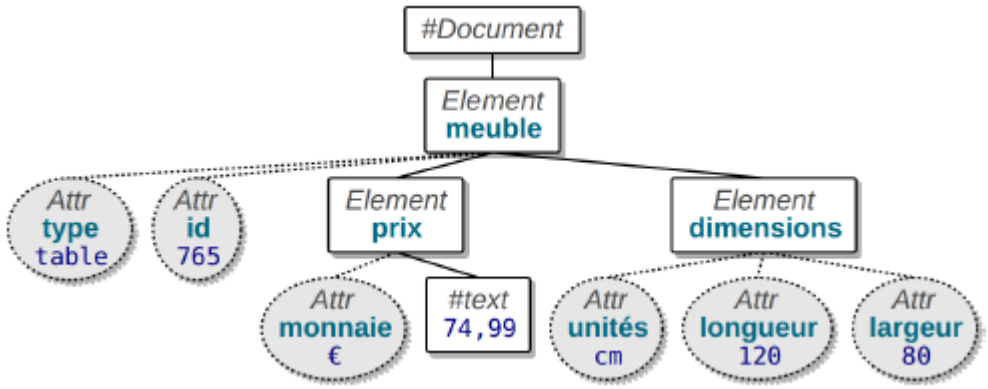
9.9.5.3 Namespace par défaut

Lorsque la racine du document définit un attribut xmlns="URI", alors par défaut toutes les balises du document sont placées dans ce namespace, donc pas besoin de mettre un préfixe.

```
<?xml version="1.0" encoding="utf-8"?>
<book xmlns="http://docbook.org/ns/docbook">
```

Ça peut s'appliquer également localement à un élément et ça concerne toute sa descendance :

```
<meuble id="765" xmlns="urn:iutlan:meubles">
  <table prix="74,99€">acajou</table>
  <table border="1" xmlns="http://www.w3.org">
    <tr><th>longueur</th>...</tr>
    <tr><td>120cm</td>...</tr>
 </table>
</meuble>
```



9.9.5.4 Attributs

Les attributs caractérisent un élément. Ce sont des couples nom="valeur" ou nom='valeur'. Ils sont placés dans la balise ouvrante.

```
<?xml version="1.0" encoding="utf-8"?>
<meuble id="765" type='table'>
        <prix monnaie='€'>74,99</prix>
        <dimensions unites="cm" longueur="120" largeur="80"/>
        <description langue='fr'>Belle petite table</description>
</meuble>
```

Remarques :

- Il n'y a pas d'ordre entre les attributs d'un élément,
- Un attribut ne peut être présent qu'une fois.

Chapitre 10 : Le fichier JSON

10.1 Définitions

JSON (JavaScript Object Notation) est un format d'échange de données léger et donc performant. C'est un format de texte indépendant de tout langage mais utilisant des conventions familières aux programmeurs de la famille de langages C (incluant JavaScript et Python notamment).

JSON est une syntaxe pour sérialiser* des objets, tableaux, nombres, chaînes de caractères, booléens et valeurs null. Elle est basée sur la syntaxe de JavaScript mais en est distincte : du code JavaScript n'est pas nécessairement du JSON, et du JSON n'est pas nécessairement du JavaScript.

*Sérialiser = mettre des données en série après les avoir converti dans un format donné. Par extension, la sérialisation est en informatique l'action de mettre des données sous forme binaire et de les écrire dans un fichier.

JSON peut représenter des nombres, des booléens, des chaînes, la valeur null, des tableaux (séquences de valeurs ordonnées) et des objets constitués de ces valeurs (ou d'autres tableaux et objets). JSON ne représente pas nativement des types de données plus complexes tels que des fonctions, des expressions régulières, des dates, etc.

Tout comme XML, JSON a la capacité de stocker des données hiérarchiques contrairement au format CSV plus traditionnel.

10.2 Structures de données et leur représentation JSON

JSON est construit par rapport à deux structures :

- Une collection de paires nom / valeur. Dans les différents langages, ce type de structure peut s'appeler objet, enregistrement, dictionnaire, table de hachage, liste à clé ou tableau associatif.
- Une liste ordonnée de valeurs. Dans la plupart des langages, c'est ce qu'on va appeler tableau, liste, vecteur ou séquence.

Ces deux structures sont des structures de données universelles. Pratiquement tous les langages de programmation modernes les prennent en charge sous une forme ou une autre. Il est logique qu'un format de données interchangeable avec les langages de programmation soit également basé sur ces structures.

En JSON, ces deux structures se retrouvent sous les formes suivantes :

- Un objet est un ensemble non ordonnées de paires nom : valeur. Un objet commence avec { et se termine avec }. Les noms sont suivis de : et les paires nom : valeur sont séparées par des ,
- Un tableau est une collection ordonnée de valeurs. Un tableau commence avec [et se termine avec]. Les valeurs sont séparées par des ,

10.3 Exemple de fichier JSON

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    },
    {
      "name": "Madame Uppercut",
      "age": 39,
      "secretIdentity": "Jane Wilson",
      "powers": [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
      "secretIdentity": "Unknown",
      "powers": [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```

10.4 Selon certain langage

JavaScript Object Notation est un format texte qui permet de représenter des objets complexes (biblio).

```
class Article {
   private int id;
   private String nom;
   private float prix;
   private String[] infos;
}
```

```
{
  "id": 1,
  "nom": "ballon de basket",
  "prix": 12.50,
  "infos": ["certifié", "orange", "renforcé"]
}
```

10.5 Avantages

	Avantages
XML	XML est extensible XML permet de définir une structure particulière pour chaque document Possibilité de définir une description du document (DTD ou schéma) permettant de le valider
JSON	La vitesse de traitement. La simplicité de mise en oeuvre. Plus compacte Exploitable directement avec JavaScript

10.6 Les bases de JSON

10.6.1 JSON Data Types

JSON a pour structure suivante :

- **Name (un nom) ou une Key(clé)/value pair**
  Consiste d'une clé ou un attribut et une valeur.
- **Object (un objet)**
  Une collection en ordre de nom/valeur pairs
- **Array (un tableau)**

  Une collection de valeur en ordre.

a. Name/value pairs

Exemple 1 : *nameValue.json*

```
{
  "conference": "OSCON",
  "speechTitle": "JSON at Work",
  "track": "Web APIs"

}
```

b. Objects

Exemple 2 : *simpleJsonObject.json*

```
{
  "address" : {
    "line1" : "555 Any Street",
    "city" : "Denver",
    "stateOrProvince" : "CO",
    "zipOrPostalCode" : "80202",
    "country" : "USA"
 }
}
```

Un objet avec un tableau

```
{
   "speaker" : {
      "firstName": "Larson",
      "lastName": "Richard",
      "topics": [ "JSON", "REST", "SOA" ]
  }
}
```

Un mixe :

```
{
 "speaker" : {
   "firstName": "Larson",
   "lastName": "Richard",
   "topics": [ "JSON", "REST", "SOA" ],
   "address" : {
     "line1" : "555 Any Street",
     "city" : "Denver",
     "stateOrProvince" : "CO",
     "zipOrPostalCode" : "80202",
     "country" : "USA"
   }
 }
}
```

c. Arrays

```
    {
       "presentations": [
         {
           "title": "JSON at Work: Overview and Ecosystem",
           "length": "90 minutes",
           "abstract": [ "JSON is more than just a simple replacement for XML when",
                         "you make an AJAX call."
                       ],
           "track": "Web APIs"
    },
    {
           "title": "RESTful Security at Work",
           "length": "90 minutes",
           "abstract": [ "You've been working with RESTful Web Services for a few years",
                         "now, and you'd like to know if your services are secure."
                       ],
          "track": "Web APIs"
     }
  ]

}
```

10.6.2 JSON Values Types

JSOn Values Types représente les Données types qui sont les éléments à droite après les :.

On nomme les types suivants :

- object
- array
- string
- number
- boolean
- null

Exemple pour le String :

```
[
  "fred",
  "fred\t",
  "\b",
  "",
  "\t",
 "\u004A"
]
```

Les Strings peuvent prendre les propriétés suivantes :

- Le zéro ou plusieurs caractères unicodes entourés de " ".
- Mais un seul quote n'est pas possible ' donc non valide.

On peut utiliser les syntaxes classiques d'espace de caractères :

- \" : Double quote
- \\ : Backslash
- \/ : Forward slash
- \b : Backspace
- \f : Form feed
- \n : Newline
- \r : Carriage return
- \t : Tab
- \u : Trailed by four hex digits

Exemple pour le Number :

```
{
 "age": 29,
 "cost": 299.99,
 "temperature": -10.5,
 "unitCost": 0.2,
 "speedOfLight": 1.23e11,
 "speedOfLight2": 1.23e+11,
 "avogadro": 6.023E23,
 "avogadro2": 6.023E+23,
 "oneHundredth": 10e-3,
 "oneTenth": 10E-2
}
```

Dans JSON, il est possible utiliser les doubles, les float, les nombres exposants de 10, octave et hexadécimale, l'infini.

Exemple pour le Boolean :

```
{
  "isRegistered": true,
  "emailValidated": false
}
```

Exemple pour le null :

Mettre la valeur à null.

```
{
  "address": {
  "line1": "555 Any Street",
  "line2": null,
  "city": "Denver",
  "stateOrProvince": "CO",
  "zipOrPostalCode": "80202",
  "country": "USA"
}
```

Chapitre 11 : Le fichier YAML

11.1 Présentations

YAML, acronyme de **Yet Another Markup Language** dans sa version 1.0, il devient l'acronyme récursif de **YAML Ain't Markup Language** (*« YAML n'est pas un langage de balisage »*) dans sa version 1.1, est un format de représentation de données par sérialisation Unicode. Il reprend des concepts d'autres langages comme XML ou encore du format de message électronique tel que documenté par RFC 2822. YAML a été proposé par Clark Evans en 2001, et implémenté par ses soins ainsi que par Brian Ingerson et Oren Ben-Kiki.

Son objet est de représenter des informations plus élaborées que le simple CSV en gardant cependant une lisibilité presque comparable, et bien plus grande en tout cas que du XML.

11.2 Caractéristiques

L'idée de YAML est que presque toute donnée peut être représentée par combinaison de listes, tableaux associatifs et données scalaires. YAML décrit ces formes de données (les *représentations YAML*), ainsi qu'une syntaxe pour présenter ces données sous la forme d'un flux de caractères (le *flux YAML*).

Une application informatique passe du flux YAML à la représentation YAML par l'opération de *chargement* (anglais *load*). Elle passe de la représentation au flux par l'opération de *déchargement* (anglais *dump*).

La syntaxe du flux YAML est relativement simple, efficace, moins verbeuse que du XML, moins compacte cependant que du CSV. Elle a été établie pour être le plus lisible possible par des humains, tout en pouvant être mise en correspondance facilement avec les types de données précités, communs dans les langages de haut niveau. À ces langages il emprunte certaines notations.

- Les commentaires sont signalés par le signe dièse (#) et se prolongent sur toute la ligne. Si par contre le dièse apparait dans une chaine, il signifie alors un nombre littéral.
- Une valeur nulle s'écrit avec le caractère tilde (~)
- Il est possible d'inclure une syntaxe JSON dans une syntaxe YAML.
- Les éléments de listes sont dénotés par le tiret (-), suivi d'une espace, à raison d'un élément par ligne.
- Les tableaux sont de la forme *clé: valeur*, à raison d'un couple par ligne.
- Les scalaires peuvent être entourés de guillemets doubles ("), ou simples ('), sachant qu'un guillemet s'échappe avec un antislash (\), alors qu'une apostrophe s'échappe avec une autre apostrophe[4]. Ils peuvent de plus être représentés par un bloc indenté avec des modificateurs facultatifs pour conserver (|) ou éliminer (>) les retours à la ligne.
- Plusieurs documents rassemblés dans un seul fichier sont séparés par trois traits d'union (---) ; trois points (...) optionnels marquent la fin d'un document dans un fichier.
- Les nœuds répétés sont initialement signalés par une esperluette (&) puis sont référencés avec un astérisque (*) ; JSON, un langage concurrent de YAML, est compatible avec la syntaxe de JavaScript mais ne supporte pas cette notion de référence.

- L'indentation, par des espaces, manifeste une arborescence.

Il est aussi possible de préciser le type (anglais *tag*) d'une donnée. Cependant, cette précision n'opère aucune contrainte, et fonctionne plutôt comme un marquage, ou une modélisation.

Un fichier YAML est analysable en une seule passe de lecture.

La syntaxe YAML se distingue de JSON par le fait qu'il se veut plus facilement lisible par une personne. Il se distingue du XML par le fait qu'il s'intéresse d'abord à la sérialisation de données, et moins à la documentation.

phpMyAdmin permet l'export des bases MySQL en YAML, entre autres formats.

11.3 Exemple

```
---
receipt:     Oz-Ware Purchase Invoice
date:        2012-08-06
customer:
    given:   Dorothy
    family:  Gale

items:
    - part_no:   A4786
      descrip:   Water Bucket (Filled)
      price:     1.47
      quantity:  4

    - part_no:   E1628
      descrip:   High Heeled "Ruby" Slippers
      size:      8
      price:     100.27
      quantity:  1

bill-to:  &id001
    street: |
            123 Tornado Alley
            Suite 16
    city:   East Centerville
    state:  KS

ship-to:  *id001

specialDelivery:  >
    Follow the Yellow Brick
    Road to the Emerald City.
    Pay no attention to the
    man behind the curtain.
...
```

Partie 3 : la gestion de projet et modélisations

Chapitre 1 : Le développement logiciel

1.1 Le Logiciel

Il existe une grande variété de logiciels et de nombreuses manières de les classer.

Une première différenciation oppose les logiciels « génériques » ou « prologiciels », qui sont vendus en grand nombre comme des produits de consommation, aux logiciels « spécifiques » qui sont développés pour contexte et un client particulier.

Une seconde différenciation repose sur la nature de la dépendance entre les logiciels et leurs environnements. Elle distingue trois classes :

- Les logiciels « autonomes » (standalone), comme les traitements de texte, les logiciels de calcul ou les jeux,
- Les logiciels qui gèrent des processus (process support), aussi bien industriels que de gestion,
- Les logiciels « embarqués » dans des systèmes (embedded), comme dans les transports, la téléphonie ou les satellites.

Dans cette dernière classe, on a coutume de parler plutôt de « développement système » que de « développement logiciel », car les problématiques matérielles et logicielles y sont indissociables.

Une troisième différenciation sépare les logiciels isolés des « lignes de produits logicielles ». Celles-ci regroupent une famille de logiciels présentant de fortes similarités et un même domaine d'application, comme des applications pour téléphones mobiles. Elles sont développées à partir d'éléments réutilisables (assets) que peuvent être des exigences, des modèles, des codes (composant), des plans de tests, etc.

Dans les grandes organisations, on particularise souvent ce qu'on appelle les « logiciels patrimoine » (legacy software). Ce sont les applications les plus anciennes, complexes et peu documentées, auxquelles on touche le moins possible tant qu'elles remplissent correctement leurs fonctions en général vitales pour l'organisation.

Enfin les « logiciels web », ou « application web », sont souvent considérés comme une catégorie à part. Ce sont des logiciels hébergés sur un serveur web et manipulables via les réseaux, Internet ou réseaux locaux, grâce à un navigateur web. On peut citer comme particularités notables :

- Les accès concurrents qu'ils subissent, avec une charge difficilement prédictible,
- Leurs exigences élevées de performance, de disponibilité et de sécurité,
- La prédominance des données par rapport aux traitements,
- Leurs besoins fréquents d'évolution,
- L'importance de l'ergonomie et l'esthétique dans leur conception.

1.2 La complexité du logiciel

Techniquement, le terme « logiciel »désigne un riche ensemble d'artefacts incluant :

- Des codes sources et des exécutables,
- Des programmes et des données de test,
- Des fichiers de configuration,
- Des documentations « externes » à destination des utilisateurs,
- Des documentations « internes » à destination de l'équipage de développement, etc.

1.3 Le développement logiciel

Le logiciel peut être développé dans plusieurs contextes assez différents.

- Chez les éditeurs de logiciels qui commercialisent les prologiciels qu'ils développent.
- Dans des Sociétés de service en ingénierie informatique (SSII), également nommées Entreprise de services du numérique (ESN), qui répondent aux besoins d'externalisation des projets informatiques, en développant des logiciels pour le compte de leurs clients.
- En interne, dans des organisations de toutes natures, entreprises, administrations et associations.
- Sur Internet, au moins en partie par des développeurs bénévoles, pour les logiciels libres.

1.4 Les activités du développement

Schématiquement, le développement logiciel consiste à transformer une idée ou un besoin en un logiciel fonctionnel. L'idée est produite par un client ou maître d'ouvrage (MOA). Le logiciel correspondant est développé par un fournisseur ou maître d'œuvre (MOE), puis exploité par des utilisateurs finaux. Lorsqu'il s'agit d'entités on parle de Maîtrise d'ouvrage (MOA) et le Maître d'œuvre (MOE). La MOA peut être assistée en externe et en interne, souvent par d'anciens informaticiens ayant une bonne pratique de la conduite des projets : on parle de MOAD, pour maîtrise d'ouvrage déléguée.

Le développement de logiciel comprend en amont une phase centrée sur les besoins (ou exigences). Il s'agit pour toutes les parties prenantes d'un projet informatique de s'accorder sur l'ensemble des besoins auxquels devra répondre le système ou logiciel à construire. Le terme d' « ingénierie des

exigences » (requirements engineering) caractérise toutes les activités liées à la gestion des besoins, y compris celles qui se poursuivent tout au long du développement, comme la gestion des évolutions des besoins et de leur traçabilité.

1.5 La qualité du logiciel

La notion de « qualité du logiciel » n'est pas simple à définir car elle est multiforme et dépend du point de vue adopté, client ou fournisseur. Pour le client, le logiciel doit répondre à ses besoins (adéquation), être efficace, facile d'apprentissage et d'utilisation (ergonomie), être fiable, robuste et sûr, être peu coûteux et livré dans les délais. Pour le fournisseur, le coût et la durée de développement sont généralement primordiaux, de même que la facilité d'extension de maintenance, d'adaptation et d'évolution, ainsi que la portabilité et l'interopérabilité.

Il est important de ne pas confondre les termes qui caractérisent les différentes formes de qualité.

La signification des plus courants d'entre eux est rappelée dans le tableau suivant :

Terme français	Terme anglais	Signification
Disponibilité	Availability	Capacité à délivrer le service attendu
Fiabilité	Reliability	Capacité à maintenir la continuité du service attendu
Sécurité	Security	Absence de conséquences catastrophiques
Sûreté	Safety	Protection contre les actions illicites
Intégrité	Integrity	Absence de d'altération inappropriées
Confidentialité	Confidentiality	Absence de divulgations non autorisées
Maintenabilité	Maintainablity	Aptitude aux réparations et aux évolutions

1.6 Principes de conception

Le **principe de séparation des responsabilités** (*separation of concerns*) vise à organiser un logiciel en plusieurs sous-parties, chacune ayant une responsabilité bien définie.

Ainsi construite de manière modulaire, l'application sera plus facile à comprendre et à faire évoluer. Au moment où un nouveau besoin se fera sentir, il suffira d'intervenir sur la ou les sous-partie(s) concernée(s). Le reste de l'application sera inchangée : cela limite les tests à effectuer et le risque d'erreur. Une construction modulaire encourage également la réutilisation de certaines parties de l'application.

Le **principe de responsabilité unique** (*single responsibility principle*) stipule quant à lui que chaque sous-partie atomique d'uun logiciel (exemple : une classe) doit avoir une unique responsabilité (une raison de changer) ou bien être elle-même décomposée en sous-parties.
"A class should have only one reason to change" (Robert C. Martin).

Exemples d'applications de ces deux principes :

- Une sous-partie qui s'occupe des affichages à l'écran participe ne devrait pas comporter de traitements métier, ni de code en rapport avec l'accès aux données.

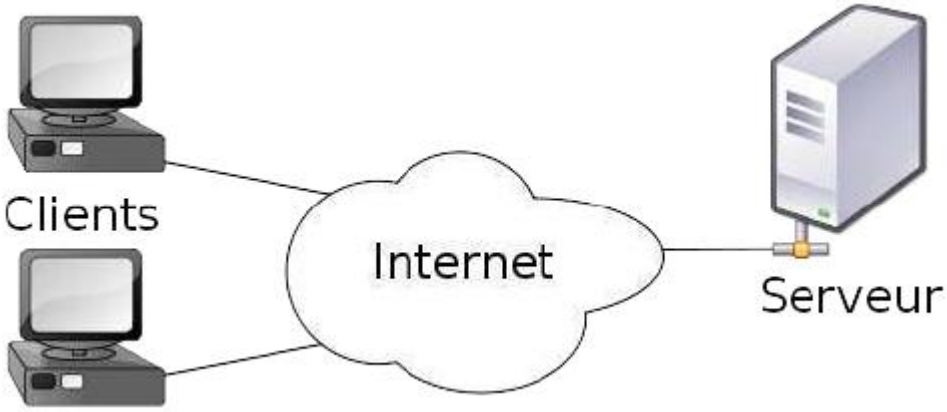
- Un composant de traitements métier (calcul scientifique ou financier, etc) ne doit pas s'intéresser ni à l'affichage des données qu'il manipule, ni à leur stockage.
- Une classe d'accès à une base de données (connexion, exécution de requêtes) ne devrait faire ni traitements métier, ni affichage des informations.
- Une classe qui aurait deux raisons de changer devrait être scindée en deux classes distinctes.

1.7 Patrons logiciels

Au fil du temps et des projets, plusieurs architectures-types se sont dégagées. Elles constituent des patrons d'architecture (*architecture patterns*) qui ont fait leurs preuves et peuvent servir d'inspiration pour un nouveau projet.

1.7.1 Architecture client/serveur

L'architecture client/serveur caractérise un système basé sur des échanges réseau entre des clients et un serveur centralisé, lieu de stockage des données de l'application.



Le principal avantage de l'architecture client/serveur tient à la centralisation des données.
Stockées à un seul endroit, elles sont plus faciles à sauvegarder et à sécuriser. Le serveur qui les héberge peut être dimensionné pour pouvoir héberger le volume de données nécessaire et répondre aux sollicitations de nombreux clients. Cette médaille a son revers :

Le serveur constitue le nœud central du système et représente son maillon faible. En cas de défaillance (surcharge, indisponibilité, problème réseau), les clients ne peuvent plus fonctionner.

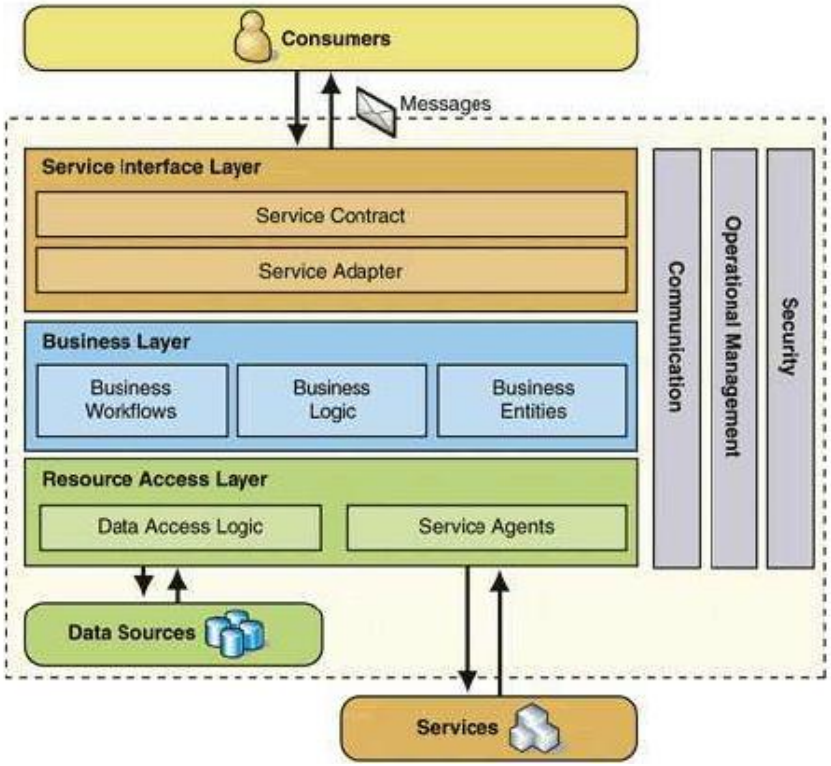
On peut classer les clients d'une architecture client/serveur en plusieurs types :

- **Client léger**, destiné uniquement à l'affichage (exemple : navigateur web).
- **Client lourd**, application native spécialement conçue pour communiquer avec le serveur (exemple : application mobile).
- **Client riche** combinant les avantages des clients légers et lourds (exemple : navigateur web utilisant des technologies évoluées pour offrir une expérience utilisateur proche de celle d'une application native).

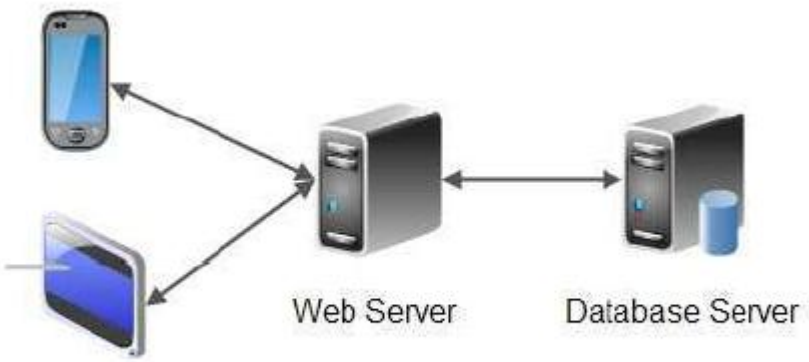
Le fonctionnement en mode client/serveur est très souvent utilisé en informatique. Un réseau Windows organisé en domaine, la consultation d'une page hébergée par un serveur Web ou le téléchargement d'une application mobile depuis un magasin central (App Store, Google Play) en constituent des exemples.

1.7.2 Architecture couches

Une architecture en couches organise un logiciel sous forme de couches (*layers*). Chaque couche ne peut communiquer qu'avec les couches adjacentes.

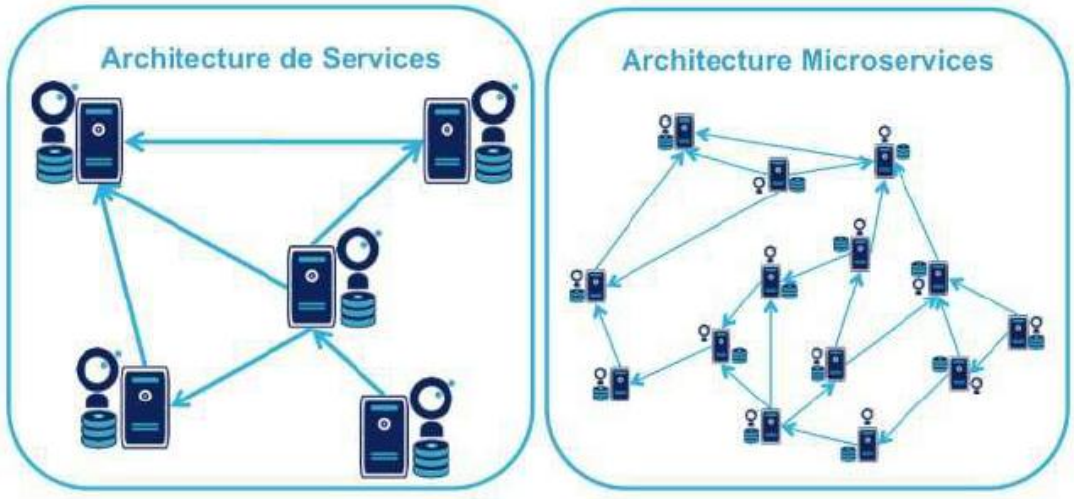


Cette architecture respecte le principe de séparation des responsabilités et facilite la compréhension des échanges au sein de l'application.
Lorsque chaque couche correspond à un processus distinct sur une machine, on parle d'architecture **n-tiers**, n désignant le nombre de couches.
Un navigateur web accédant à des pages dynamiques intégrant des informations stockées dans une base de données constitue un exemple classique d'architecture 3-tiers.



1.7.3 Architecture orientée services

Une architecture orientée services (SOA, *Service-Oriented Architecture*) décompose un logiciel sous la forme d'un ensemble de services métier utilisant un format d'échange commun, généralement XML ou JSON.
Une variante récente, l'architecture microservices, diminue la granularité des services pour leur assurer souplesse et capacité à évoluer, au prix d'une plus grande distribution du système. L'image ci-dessous (source) illustre la différence entre ces deux approches.



1.7.4 Architecture Modèle-Vue-Contrôleur

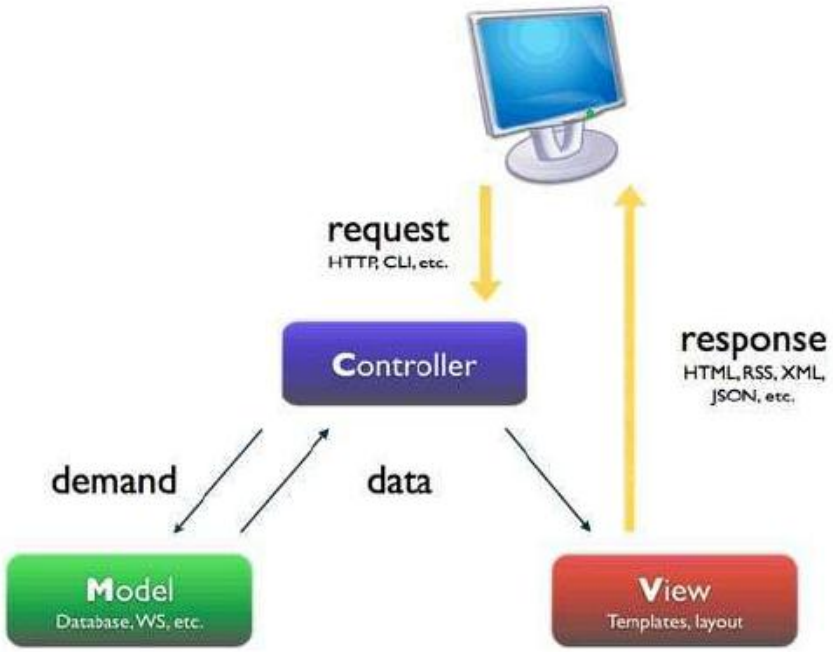
Le patron Modèle-Vue-Contrôleur, ou **MVC**, décompose une application en trois sous parties :

- La partie **Modèle** qui regroupe le logique métier ("business logic") ainsi que l'accès aux données. Il peut s'agir d'un ensemble de fonctions (Modèle procédural) ou de classes (Modèle orienté objet). ;

- La partie **Vue** qui s'occupe des interactions avec l'utilisateur : présentation, saisie et validation des données ;
- La partie **Contrôleur** qui gère la dynamique de l'application. Elle fait le lien entre les deux autres parties.

Ce patron a été imaginé à la fin des années 1970 pour le langage Smalltalk afin de bien séparer le code de l'interface graphique de la logique applicative. On le retrouve dans de très nombreux langages : bibliothèques Swing et Model 2 (JSP) de Java, frameworks PHP, ASP.NET MVC, etc.

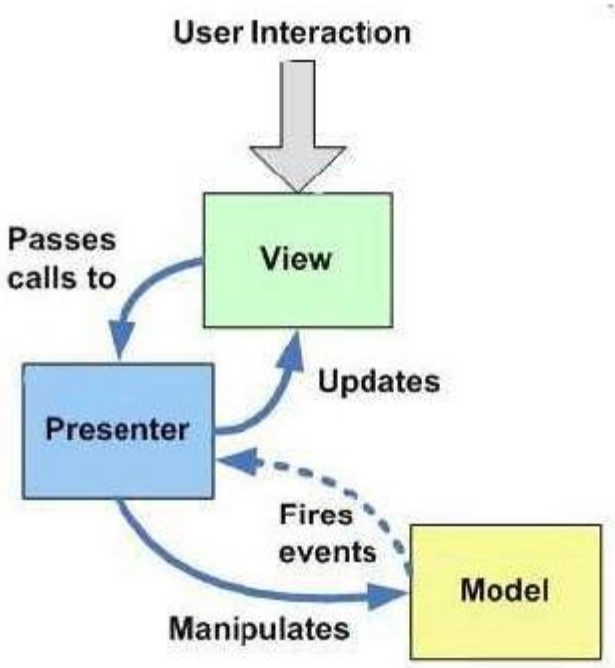
Le diagramme ci-dessous (extrait de la documentation du framework PHP Symfony) résume les relations entre les composants d'une architecture web MVC.



1.7.5 Architecture Modèle-Vue-Présentation

Le patron Modèle-Vue-Présentation, ou **MVP**, est un proche cousin du patron MVC surtout utilisé pour construire des interfaces utilisateurs (UI).

Dans une architecture MVP, la partie **Vue** reçoit les évènements provenant de l'utilisateur et délègue leur gestion à la partie **Présentation**. Celle-ci utilise les services de la partie **Modèle** puis met à jour la **Vue**.



Dans la variante dite *Passive View* de cette architecture, la Vue est passive et dépend totalement du contrôleur pour ses mises à jour. Dans la variante dite *Supervising Controller*, Vuet et Modèle sont couplées et les modifications du Modèle déclenchent la mise à jour de la Vue.

Chapitre 2 : Les modèles de développement

Un ensemble structuré d'activités nécessaires pour développer un logiciel.

Un modèle de développement de logiciel est une représentation abstraite d'un processus.

De nombreux modèles différents mais pour tous :

- **Spécification** : on définit ce que le système devra faire.
- **Conception** et implémentation : on définit l'organisation du système et on l'implémente.
- **Validation** : on vérifie que le système fait bien ce que veut le client.
- **Evolution** : on modifie le système en réponse aux changements des besoins du client.

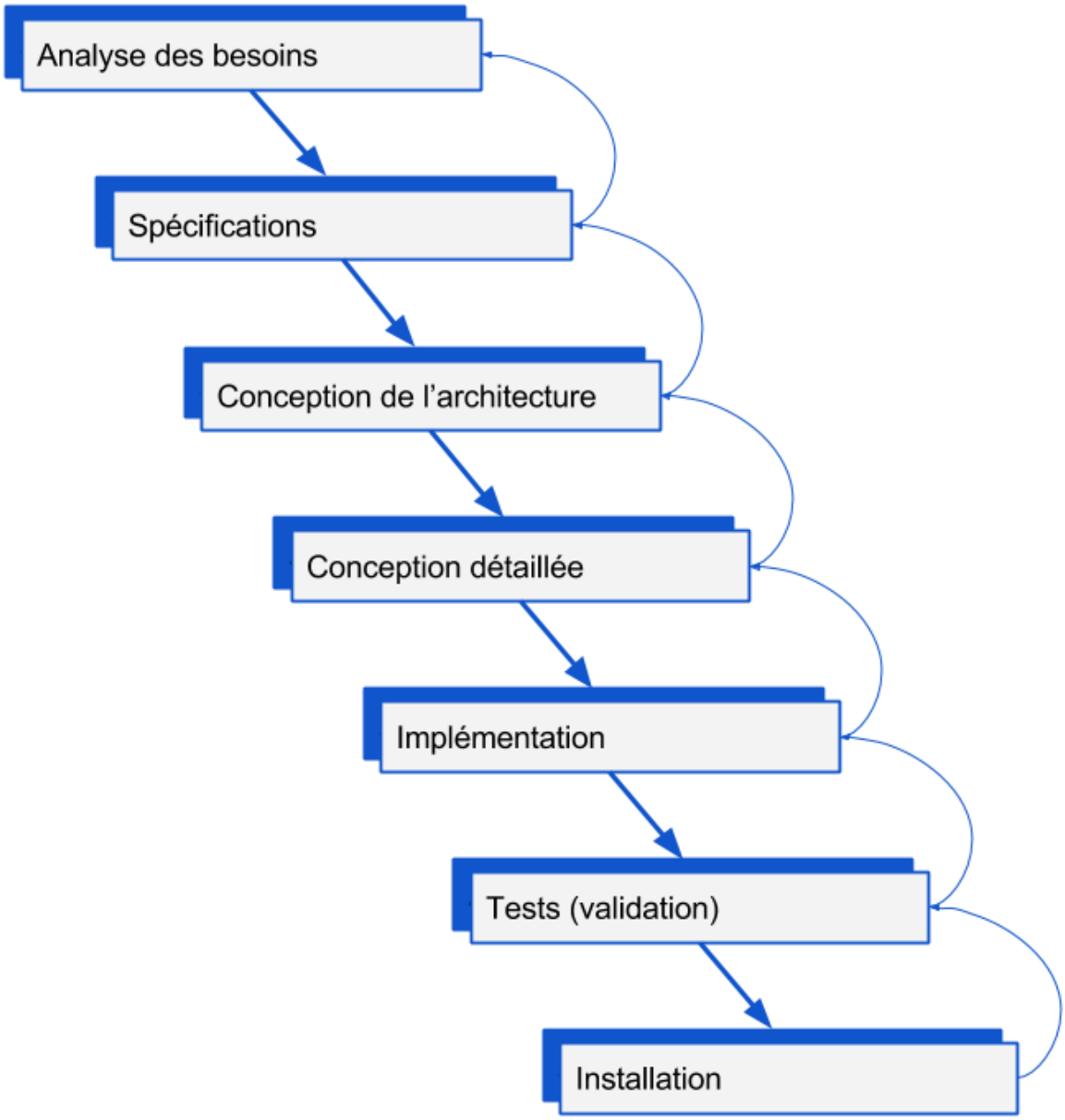
2.1 Les modèles linéaires

2.1.1. La cascade (waterfall)

Présenté par Winston W. Royce en 1970, le modèle en cascade originel est hérité du BTP. Il se base sur 2 idées fondamentales :

- Une étape ne peut pas être débutée avant que la précédente ne soit achevée : inutile de monter les murs tant que les fondations ne sont pas coulées (ça arrive parfois mais nous ne sommes pas dans une émission de défense des droits du consommateur…) ;
- La modification d'une étape du projet a un impact important sur les étapes suivantes.

Ce modèle comporte 7 phases : analyse des besoins, spécifications, conception de l'architecture, conception détaillée, implémentation, tests (validation) et enfin installation. Chacune de ces phases doit produire un ou plusieurs livrables définis à l'avance et a une date d'échéance fixée. On ne peut passer d'une étape à l'autre que lorsque les livrables de l'étape en cours sont jugés satisfaisants. Si tout se passe bien on passe à la phase suivante, sinon on remonte à la phase précédente, voire même en début de cycle si une anomalie critique est détectée.

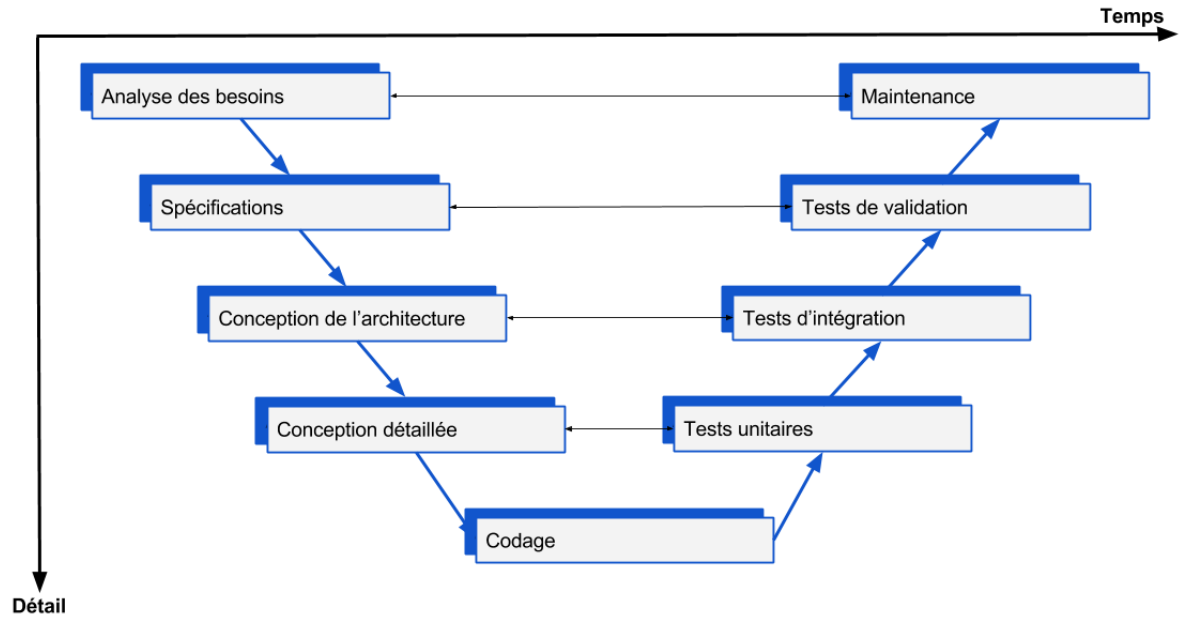


Avantages du modèle en cascade : Le planning est établi à l'avance et le maître d'ouvrage sait précisément ce qui va lui être livré et quand il pourra en prendre livraison.

Inconvénients du modèle en cascade : ils sont assez nombreux mais le principal inconvénient est la très faible tolérance à l'erreur (les anomalies sont détectées tardivement) qui induit automatiquement un coût important en cas d'anomalie.

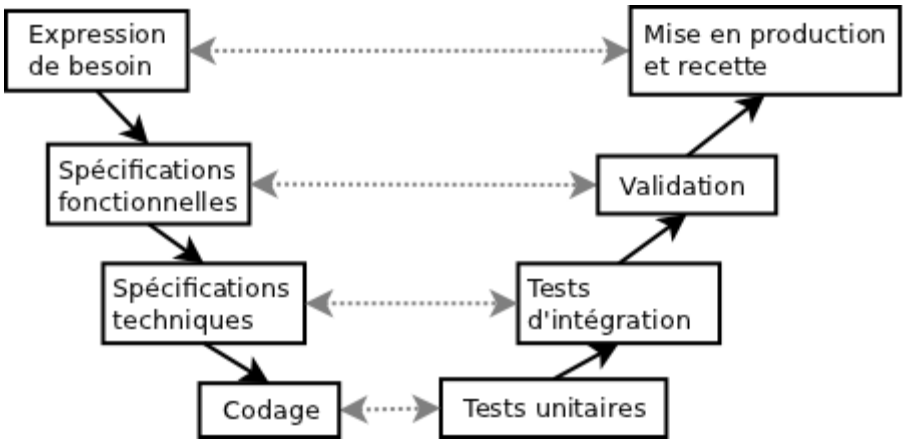
2.1.2. Le modèle en V

Face aux problèmes de réactivité que pose l'approche en cascade, l'industrie informatique a adopté le cycle en v dans les années 80. Ce modèle ne se découpe plus en 7 mais en 9 phases qui se répondent 2 à 2 : à **chaque étape de conception correspond une phase de test** ou de validation, comme vous pouvez le voir ci-dessous **:**



Le cycle en V est une méthode d'organisation très connue dont l'origine remonte à l'industrie et qui a été adaptée à l'informatique dans les années 80. C'est l'une des premières méthodes qu'on apprend à l'école après le cycle en cascade, et elle reste toujours d'actualité.

La grande force du cycle en V, c'est qu'il définit assez précisément la manière dont les choses *devraient* se passer.



On peut y distinguer 3 grandes parties : La phase de conception, la phase de réalisation (codage) et la phase de validation. Les phases de conception et de validation se découpent en plusieurs parties. Chaque étape ne peut être réalisée qu'une fois que l'étape précédente est terminée, ce qui diminue les prises de risque sur le projet.
Ce qui est bien visible sur le diagramme, c'est que chaque étape de conception possède son alter ego de validation. Il devient alors assez aisé de valider un projet, car le référentiel de test est connu très précisément.

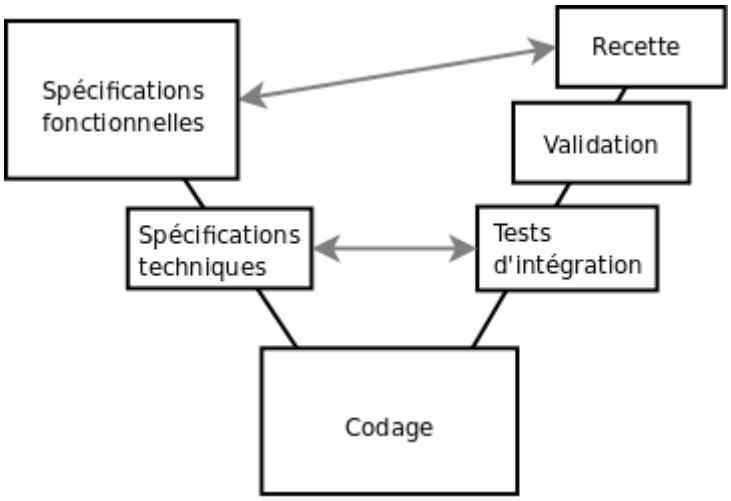
Les différentes étapes

Le cycle en V est constitué de 8 étapes qui ont toutes leur importance.

- **Expression de besoin** : Le client exprime son besoin, en décrivant les usages correspondant au produit fini tel qu'il peut l'imaginer. Cela doit répondre aux questions « Que veut-on ? » et « À quel coût ? ».
- **Spécifications fonctionnelles** : C'est le cahier des charges exact du produit final, tel que le désire le client. Il doit couvrir l'intégralité des cas d'utilisation du produit, en expliquant ce qu'il *doit faire* et non pas *comment il va le faire*.
- **Spécifications techniques** : C'est une traduction des spécifications fonctionnelles en termes techniques. C'est durant l'élaboration des specs techniques que sont choisies les technologies à mettre en œuvre pour développer le produit, et qu'est conçue l'architecture logicielle du produit.
- **Codage** : C'est la phase de réalisation à proprement parler, pendant laquelle sont développées des briques qui sont ensuite assemblées pour créer le produit fini.
- **Tests unitaires** : Ces tests interviennent à un niveau « atomique ». Chaque brique logicielle a été modélisée puis codée durant les étapes précédentes. Les tests unitaires assurent que ces briques respectent de manière individuelle leur cahier des charges.
- **Tests d'intégration** : Ce sont là les premiers tests grandeur nature du produit fini. On s'assure qu'il suit les indications des spécifications techniques.
- **Validation** : Le produit est à ce moment testé en regard de la spécification fonctionnelle. Toutes les utilisations qui y ont été définies doivent pouvoir se vérifier dans les faits.
- **Mise en production et recette** : Le produit est vérifié une dernière fois en pré-production, avant d'être *mis en production*. Le client procède à la recette, pour vérifier que son expression de besoin est respectée.

La pratique

Malheureusement, si le cycle en V est limpide d'un point de vue théorique, son application réelle est très difficile. Dans une grande majorité de cas, on voit des organisations qui ressemblent plutôt à ce schéma :



- La phase de conception se réduit à 2 étapes.
    - Les spécifications fonctionnelles, qui représentent l'ensemble des besoins du client et/ou définissent ce que doit faire le produit fini.
    - Les spécifications techniques, qui détaillent comment le produit va être réalisé techniquement.
- La phase de validation contient juste 3 étapes.
    - Les tests d'intégration, pendant lesquels on vérifie que l'intégralité du produit est valide techniquement.
    - Les tests de validation, qui sont un mélange de tests techniques et fonctionnels, et sur lesquels le client se base souvent pour décider du lancement du produit.
    - La recette, qui est utilisée pour vérifier que le produit est valide par rapport aux spécifications fonctionnelles, mais qui a tendance à n'intervenir qu'*après* la mise en production (ou bien elle est tronquée en pré-production, ce qui aboutit à mettre des bugs en production).

Nos neufs phases du cycle en V sont arrivent donc comme ceci :

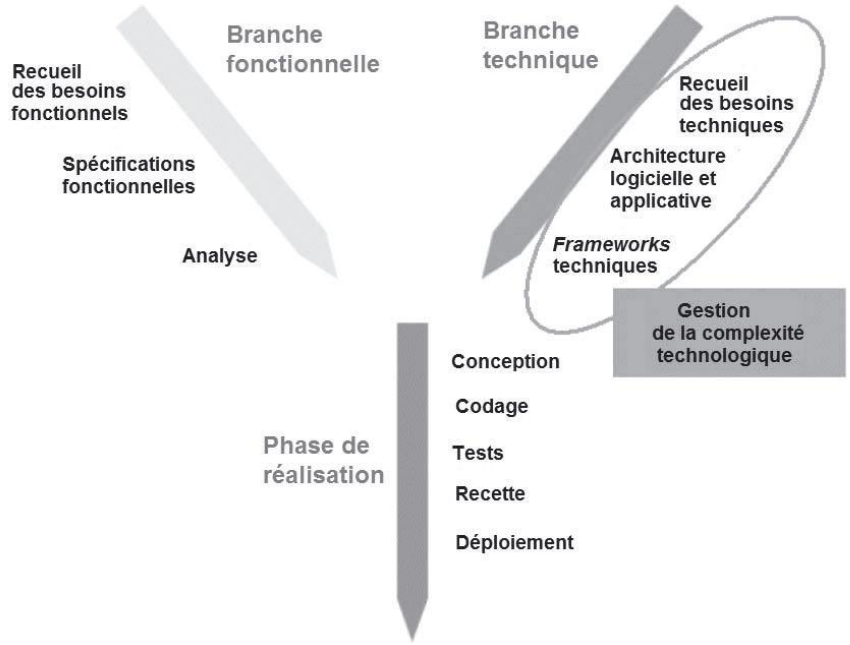
- **Etude et analyse :** l'analyse ou la définition des besoins, la rédaction des spécifications, la conception architecturale, la conception détaillées
- **Codage :** développement de l'application
- **Tests et validations :** tests unitaires, test d'intégration, tests de validation et maintenance corrective.

Avantages du modèle en V : La stricte structure en V permet d'espérer que le livrable final sera parfait, puisque les étapes de test sont aussi nombreuses que les étapes de réflexion. De plus, il est facile de prévoir les tests à réaliser au moment où l'on conçoit une fonctionnalité ou une interface, le travail s'enchaîne donc de façon assez naturelle.

Inconvénients du modèle en V : Malheureusement ce modèle est rarement utilisé tel quel et le V est bien souvent déséquilibré, tantôt côté analyse, tantôt côté recette et la marge d'erreur est bien souvent proportionnelle à la marge de liberté prise par rapport au modèle théorique.

2.1.3. Le modèle en Y

Il s'agit d'une autre variante du modèle en cascade qui distingue initialement une branche fonctionnelle et une branche technique afin de paralléliser la résolution des questions correspondantes.



2.2 Les modèles centrés su des prototypages

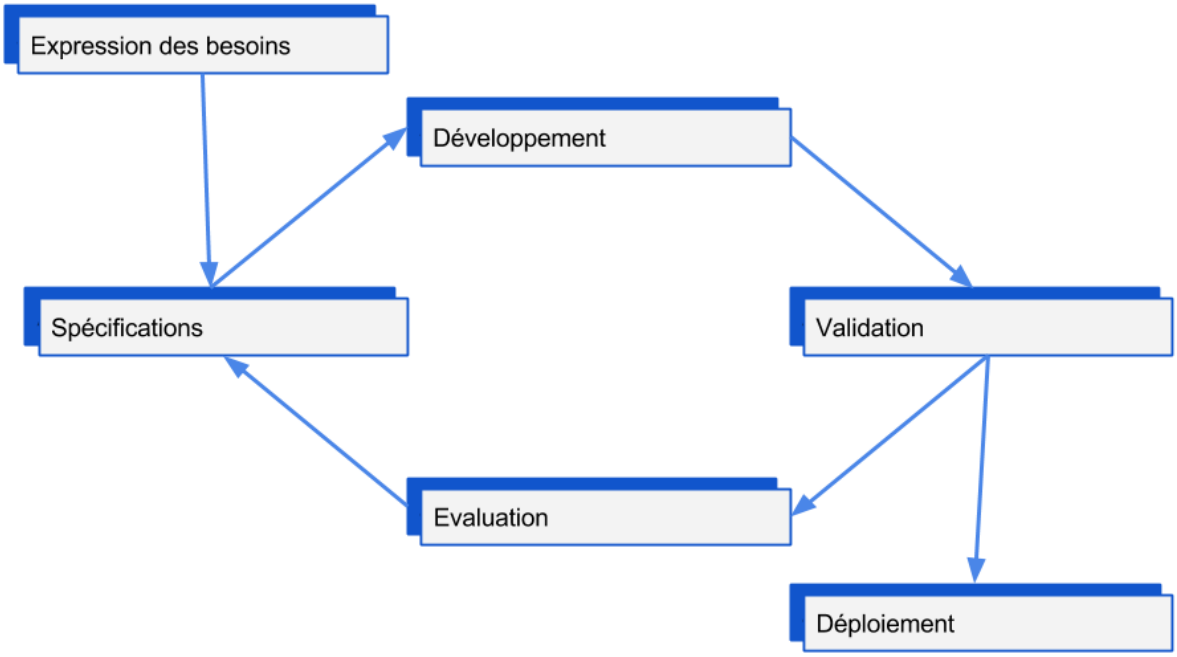
L'approche habituelle, qualifiée de « prototypage rapide », s'appuie sur le développement rapide d'un prototype avec le client pour analyser ses besoins. Le client donne son *feedback* qui conduit à des adaptations successives du prototype. Le développement du logiciel final se fait ensuite de manière classique, à partir du cahier des charges qui reprend les besoins mis en évidence via le prototype.

L'idée est de permettre la validation des spécifications par expérimentation : « Je saurai ce que je veux lorsque je le verrai ! ». Cette approche permet une validation beaucoup plus concrète des besoins, ce qui minimise les risques d'erreurs à ce niveau. Par contre, la planification initiale du projet est délicate à faire.

2.3 Les modèles itératifs et incrémentaux

2.3.1. Définition

Simplifions un peu le modèle précédent en réduisant le nombre d'étapes du cycle et **séparons les activités des artéfacts** (c'est à dire les produits issus de ces activités). Nous arrivons logiquement au modèle itératif, que vous pouvez voir ci-dessous.



Tout commence par l'**expression de besoin** : le client explique ce qu'il veut, tout en sachant que ces besoins peuvent être modifiés par la suite du processus. Ensuite, on se lance dans le processus itératif en lui-même avec la **rédaction des spécifications** qui est suivie par le **développement**, puis la **validation** (c'est à dire les tests) et enfin une **évaluation du travail** qui servira d'information de départ pour le cycle suivant en dressant le bilan des difficultés rencontrées et des fonctionnalités abandonnées pendant le cycle. A l'issue de la validation on passe aussi au **déploiement** : les livrables (il peut s'agir d »une version du site ou logiciel, ou même d'une documentation) qui ont été validés sont déployés, c'est à dire mis à disposition.

Avantages du modèle itératif : Ce type de cycle de développement est le plus souple de tous ceux présentés ici : chaque itération permet de s'adapter à ce qui a été appris dans les itérations précédentes et le projet fini peut varier du besoin qui a été exprimé à l'origine. Comme dans le cycle en spirale, la mise à disposition de livrables à chaque cycle permet un apprentissage de l'utilisateur final en douceur.

Inconvénients du modèle itératif : A mes yeux le plus gros piège de ce modèle de développement c'est la confiance qui amène bien souvent à négliger les tests d'intégration. Ainsi les développeurs livrent une nouvelle fonctionnalité sans se rendre compte qu'ils ont cassé une chose qui fonctionnait dans les cycles précédents. Il faut donc que le chef de projet soit particulièrement vigilant lors de la phase de tests.

2.3.2. Le modèle en spirale

Défini par Barry Boehm en 1988 dans son article « A Spiral Model of Software Development and Enhancement », **le cycle en spirale reprend les étapes du cycle en V**, mais prévoit l'implémentation de **versions successives**, ce qui permet de mettre l'accent sur la **gestion des risques**, la première phase de chaque itération étant dédiée à ce poste. A ce point il est nécessaire de définir la notion de **prototype**. En effet, on ne fait pas des versions successives d'un même produit fini, corriger une liste de bugs permet de passer de la béta à la finale mais pas de la v1 à la v2… Le cycle en spirale prévoit donc la livraison de prototypes, c'est à dire de versions incomplètes du produit. Il peut s'agir d'une simple maquette ou même des wireframes sans aucune fonctionnalité (on parle alors de **prototype horizontal**) ou bien de sites partiellement fonctionnels : telle version implémentera la navigation de base, la suivante ajoutera l'espace membres, puis la zone de téléchargements… on parlera alors de **prototype vertical**.

Avantages du modèle en spirale : Le but premier de ce modèle étant la gestion des risques, ceux-ci sont logiquement limités. L'expertise du client croit à chaque itération du cycle, l'apprentissage se fait par touche et pas d'un seul bloc. Enfin, ce modèle est très adaptatif : si chaque prototype apporte des fonctionnalités indépendantes, il est possible de changer l'ordre de livraison des versions.

Inconvénients du modèle en spirale : Selon moi le principal défaut du cycle en spirale c'est qu'il n'est adapté qu'aux projets suffisamment gros, inutiles de prévoir la livraison de 5 ou 6 prototypes pour un site vitrine sous WordPress (même si on peut considérer qu'une maquette puis le site en lui-même constituent 2 cycles complets). De plus l'évaluation des risques en elle-même et la stricte application du cycle de développement peut engendrer plus de coûts que la réalisation du logiciel. Enfin, ce type de cycle de développement est complexe, entre les étapes prévues en théorie et celles mises en pratique il y a une grande différence.

2.4 Les modèles agiles

2.4.1 Définition

Les **méthodes agiles** sont des groupes de pratiques de pilotage et de réalisation de projets. Elles ont pour origine le manifeste Agile, rédigé en 2001, qui consacre le terme d'« agile » pour référencer de multiples méthodes existantes.

Les méthodes agiles se veulent plus pragmatiques que les méthodes traditionnelles, impliquent au maximum le demandeur (client) et permettent une grande réactivité à ses demandes. Elles reposent sur un cycle de développement itératif, incrémental et adaptatif et doivent respecter quatre valeurs fondamentales déclinées en douze principes desquels découlent une base de pratiques, soit communes, soit complémentaires.

Les méthodes pouvant être qualifiées d'agiles, depuis la publication du manifeste Agile, sont :

- le développement rapide d'applications (1991), de l'anglais rapid application development (RAD) ;
- la version anglaise du RAD (1995), Dynamic systems development method (DSDM) ;
- plusieurs autres méthodes, comme adaptive software development *(ASD)* ou *feature driven development* (FDD) qui reconnaissent leur parenté directe avec RAD.

Les deux approches de réingénierie agile du développement d'application ou d'amélioration continue (Lean), désormais les plus utilisées sont :

- en ce qui concerne la réingénierie immédiate des processus, la méthode Extreme programming (XP), publiée en 1999 par Kent Beck ;
- en ce qui concerne l'amélioration continue, le framework Scrum, présenté en 1995 par Ken Schwaber, publié ensuite en 2001 par lui-même et Mike Beedle.

2.4.2 Les quatre préceptes

1. Les individus et leurs interactions, plus que les processus et les outils

Il s'agit ici de mettre la communication et l'humain au centre de la démarche. La création d'un logiciel n'est pas à la charge d'équipes cloisonnées par des expertises, elle se nourrit des échanges constructifs entre tous les acteurs du projet

2. Des logiciels opérationnels, plus qu'une documentation exhaustive

Il s'agit de focaliser la plus grande partie du temps sur la production et l'outil. Attention, il n'est pas question de remettre en cause la documentation et son utilité, juste de la rendre plus rationnelle, plus légère, moins chronophage et plus simple à faire évoluer.

3. La collaboration avec les clients, plus que la négociation contractuelle

L'implication du client est un prérequis fort qui assure la réussite du projet. En effet, le client fait partie de l'organisation de manière active et il est impliqué dans toutes les phases de création de son produit. La relation contractuelle ne doit pas être remise en cause mais simplement adaptée à ce dispositif : redistribution des responsabilités, nécessité d'investissement de l'ensemble des acteurs (client compris), etc.

4. L'adaptation au changement, plus que le suivi d'un plan

C'est dans l'adaptions d'un logiciel au fil de l'eau que l'agilité prend tout son sens. Cependant, il est important de comprendre que c'est également l'un des points les plus complexes.

Etre agile ne signifie pas appliquer tous les changements sans discernement dès que le client ou les utilisateurs le souhaitent.

... Douze Principes :

Les quatre préceptes de base du Manifeste agile sont déclinés en douze grands principes qui apportent des précisions sur la manière de produire agile.

	Principe	A retenir
1	Note plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grand valeur ajoutée.	Satisfaire l'utilisateur
2	Accueillez positivement les changements de besoins, même tard dans le projet. Les processus agiles exploitent le changement pour donner un avantage compétitif au client.	Accepter le changement
3	Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.	Livrer fréquemment
4	Les utilisateurs ou leurs représentants et les développeurs doivent	Travailler en synergie

	travailler ensemble quotidiennement tout au long du projet.	
5	Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites leur confiance pour atteindre les objectifs fixés.	Stimuler la motivation
6	La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est de dialogue en face-à-face.	Communiquer en direct avec les opérationnels
7	Un logiciel opérationnel est la principale mesure d'avancement.	Un seul indicateur : les fonctionnalités implémentées
8	Les processus agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.	Bannir les rushs de production
9	Une attention continue à l'excellence technique et à la bonne conception renforce l'agilité.	Ne pas négliger la qualité de production
10	La simplicité- c'est-à-dire l'art de minimiser la quantité de travail inutile- est essentiel.	Rester concentré sur l'essentiel : mettre en place un produit
11	Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.	Favoriser une certaine autonomie des équipes
12	A intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.	Intégrer la notion d'amélioration continue tout au long du projet

2.4.3 Une multitude de méthodes

On parle des méthodes suivantes :

- **RAD** : La méthode RAD (Rapid Application Development) est considérée comme étant la première méthode agile. Elle est publiée par James Martin en 1991 et reprend de nombreux travaux orientés autour des cycles en spirales et du développement incrémental.

RAD a posé les bases des techniques de réalisation propres aux méthodes agiles. Elle propose un mode de développement dit « semi-itératif » : seule la phase de développement comprend des « itérations » permettant des ajustements à la marge, l'expression de besoin et la conception étant déterminés en début de projet.

- **DSDM** : La méthode DSDM (Dynamic Systems Development Methdod) a été mise en place au milieu des années 1990 en Grande-Bretagne. L'idée proposée par un regroupement de sociétés est alors professionnaliser et de structurer les pratiques liées à la méthode RAD.

Par rapport à RAD, l'apport principal de DSDM port deux concepts complémentaires : les neuf principes du DSDM et un processus de réalisation spécifique.

- **FDD :** La méthode FDD (Feature Driven Development) est spécifiée en 1997 par Jeff De Luca et propose une gestion de projet pilotée par les fonctionnalités à forte valeur ajoutée pour

les utilisateurs (d'où son nom). Il s'agit donc de mettre en place des cycles itératifs relativement courts et permettant de proposer des fonctionnalités tangibles.

FDD définit pour chaque itération un processus en cinq étapes allant de la conception à la réalisation. On parle ici d'itération (et non de semi-itération) puisque chacune d'elles comporte sa propre étape de conception, accompagnée de la réalisation d'un ensemble de fonctionnalités.

- **Crystal**

Crystal n'est pas a proprement parler une méthode mais plutôt un ensemble de méthodes agiles. Le concept est développé par Alistair Cockburn en 2004 sur le principe suivant : chaque projet est différent, la méthode doit pouvoir s'adapter simplement à leurs spécificités.

La méthode Crystal Clear est particulièrement adaptée aux petites équipes de développement. Idéalement, l'équipe est composée d'un architecte et de deux à six ou sept développeurs, situés à proximité les uns des autres, de façon à faciliter la communication, dans un local calme. Des tableaux blancs servent de supports afin que tous aient un accès rapide à toutes les informations. Les rythmes de développement et de livraison sont rapides (toutes les deux semaines ou une fois par mois) afin que les utilisateurs puissent passer les tests.

Durant tout le processus de développement, l'équipe se remet en question en permanence afin d'améliorer continuellement sa façon de travailler.

- **eXtreme Programming**

eXtreme Programming ou XP, fait partie des méthodes les plus populaires depuis sa sortie en 1996. Ses créateurs Kent Beck, Ward Cunningham et Ron Jeffries sont partis d'un principe simple : pousser à l'extrême les bonnes pratiques de réalisation d'un projet. Parmi celles-ci, la plus populaire concerne le pair programming (un poste pour deux développeurs) permettant de pousser au paroxysme la pratique de revue de code. Cependant, on trouve également d'autres concepts comme celui de toujours choisir la solution la plus simple ou encore celui de faire des livraisons plusieurs fois par jour si nécessaire.

- **Devops**

Philosophie en constante évolution, le DevOps offre un framework conçu pour dynamiser et améliorer le développement d'applications et accélérer la mise à disposition de nouvelles fonctionnalités, de mises à jour logicielles ou de produits.

Il favorise la communication, la collaboration, l'intégration, la visibilité et la transparence continues entre les équipes chargées du développement d'applications (Dev) et celles responsables des opérations IT (Ops).

Cette relation plus étroite entre Dev et Ops se reflète dans chaque phase du cycle de vie DevOps : planification logicielle initiale, codage, développement, test, publication, déploiement, opérations et surveillance continue. Elle génère de façon constante des retours clients, ce qui renforce le potentiel d'amélioration lors du développement, des tests et du déploiement. La publication accélérée et permanente des modifications ou ajouts de fonctionnalités en est un exemple.

Les objectifs du DevOps s'articulent autour de quatre catégories : culture, automatisation, mesure et partage. Dans chacun de ces domaines, les outils DevOps améliorent la rationalisation et la collaboration des workflows de développement et d'opérations en automatisant les tâches chronophages, manuelles ou statiques des phases d'intégration, de développement, de test, de déploiement ou de surveillance.

- **Méthode Prince 2**

- **Scrum**

La méthode agile Scrum particulièrement destinée à la gestion de projets informatiques tient son nom du monde du rugby. Le principe de Scrum est de pouvoir modifier la direction prise par le projet au fur et à mesure de son avancement. C'est exactement ce qui se passe lors d'un match de rugby, lors d'une mêlée (« scrum » en anglais).

La mêlée est donc une phase essentielle au rugby comme dans la gestion de projet. Si les conditions de réussite ne sont pas remplies, alors il faut réorienter le projet pour repartir sur de meilleures bases. Le client est étroitement impliqué grâce à la livraison régulière de prototypes opérationnels permettant de valider les développements. Cette gestion dynamique permet de s'assurer de la correspondance entre le besoin exprimé et le produit livré, et de réorienter au besoin les futurs développements.

- **LSD**

Le Lean Software Development est basé sur sept grands principes.

- Éliminer les gaspillages (finition partielle, processus inutiles, fonctionnalités non nécessaires, modification de l'équipe, retards…),
- Favoriser l'apprentissage (multiplication des sources d'apprentissage, synchronisation des équipes…),
- Reporter les décisions (jusqu'au dernier moment raisonnable, pour éviter de longues discussions sources de pertes de temps et les décisions irrévocables),
- Livrer vite (livraisons rapides et régulières de façon à avoir un retour client rapide également),
- Responsabiliser l'équipe (favoriser l'autonomie et le leadership des équipes, partir du principe que les intervenants connaissent leur travail, faciliter le développement de l'expertise),
- Construire la qualité (elle doit être placée au cœur du projet, de la conception à la réalisation),
- Optimiser le système dans son ensemble (mise en place de mesures de performances complètes, pour avoir en permanence une vision globale du produit, et gérer les différentes interactions et dépendances).

Avec la méthode Lean, la qualité est réellement placée au cœur de la gestion du projet, en optimisant notamment l'ensemble des processus d'apprentissage, de décision, de livraison et de mesure de performances.

- **Agile Unified Process (Agile UP or AUP)**

Agile Unified Process (ou Processus Unifié Agile) est une version simplifiée du Rational Unified Process, ou RUP. Il s'agit d'une méthode de développement d'applications métier utilisant les techniques agiles du TDD (Test Driven Development ou développement piloté par les tests), du MDD (Model Driven Development ou développement piloté par le modèle) et de la gestion du changement.

La méthode est divisée en quatre phases :

- Lancement : identification du périmètre du projet, définition de la ou des architectures potentielles pour le système, implication des intervenants et obtention du budget.
- Conception : définir l'architecture du système et démonstration de sa pertinence.
- Réalisation : développement du logiciel lors d'un processus incrémental dans l'ordre de priorité des fonctionnalités.
- Livraison : validation et déploiement du système en production.

- **Dynamic Systems Development Method (DSDM)**

Cette méthode s'articule autour de neufs grands principes qui sont la participation des utilisateurs, l'autonomie de l'équipe projet, la transparence des développements, l'adéquation avec le besoin, le développement itératif et incrémental, la réversibilité permanente, la synthèse du projet, les tests automatisés et continus et enfin la coopération entre tous les intervenants.

Le projet commence par une étude de faisabilité afin de décider s'il faut le faire ou non. Un rapport est rédigé, et éventuellement, un prototype est créé pour démontrer la faisabilité de l'application. S'il a été décidé de continuer, une analyse fonctionnelle est réalisée et les spécifications sont rédigées. A partir de ce moment-là, des itérations de conception technique et de développement sont mises en place, puis au final, l'application est livrée en production.

- **Adaptive software development (ASD)**

ASD est une méthode de développement rapide d'applications. Le principe consiste à automatiser et à industrialiser un maximum de processus. Des outils de modélisation sont utilisés et une usine logicielle est mise en place de façon à générer un maximum de code informatique automatiquement. Un atelier de génie logiciel (AGL) permet ensuite aux développeurs de modifier l'application ainsi générée. Pour finir, une usine de livraison assure l'automatisation de l'ensemble des processus de déploiement.

La méthode ASD est indépendante de toute méthodologie ou langage de programmation.

- **Behavior driven development(BDD)**

Dans cette méthode, c'est le langage naturel qui est mis en avant. Plutôt que de décrire les solutions techniques à mettre en œuvre, ce sont les objectifs des fonctionnalités qui sont décrites par les utilisateurs. Ces derniers sont donc particulièrement impliqués dans le processus de fabrication. Le comportement cible de l'application est donc décrit au travers d'exemples permettant aux

développeurs de mieux cerner les objectifs. Les expressions « Etant donné », « quand », « alors » et « et » sont employées dans les scénarios décrits, qui sont également utilisés pour créer des séries de tests de non régression. Les scénarios sont écrits collectivement avec le client, les développeurs et toutes les équipes impliquées dans le processus.

- **Conception pilotée par le domaine (DDD domain-driven design)**

Le Domain Driven Design est une technique de conception d'applications informatiques. La conception est centrée sur le domaine métier et non sur les aspects techniques. Elle permet aux équipes techniques et fonctionnelles de communiquer ensemble afin d'obtenir un modèle commun de l'application, compréhensible par tous. Les développeurs acquièrent ainsi une meilleure connaissance du fonctionnel et de son vocabulaire spécifique, et l'équipe métier a une meilleure vision des contraintes techniques.

Grâce à cet outil, on obtient une meilleure communication entre les différents intervenants et une conception modulaire facilitant à terme la maintenabilité de l'application et la réutilisabilité de ses composants.

- **Test driven development (TDD)**

Le développement piloté par les tests est une technique de développement qui associe l'écriture des tests unitaires, la programmation et le remaniement du code. Les tests unitaires sont écrits avant le code. Chaque test décrit un élément de la fonctionnalité. Le développeur écrit le minimum de code possible pour que le test passe, et qu'il échoue pour des raisons prévisibles. Au fur et à mesure de l'avancement, le code source doit être simplifié autant que nécessaire et continuer à passer les tests. Les tests s'accumulent durant le développement et sont exécutés automatiquement de façon très régulière.

Le TDD est toujours associé à des outils d'automatisation de tests unitaires liés au langage de programmation utilisé.

- **Rational Unified Process (RUP)**

Cette méthode est un mélange des pratiques classiques et agiles de gestion de projet. Les développements sont itératifs et incrémentaux. Chaque itération respecte un cycle comprenant quatre phases qui sont lancement, conception, réalisation et livraison. Les développements sont guidés par des cas d'utilisation. On commence par les fonctionnalités génériques pour aller ensuite de plus en plus vers le spécifique.

- **Disciplined Agile Delivery (DAD)**

DAD est une méthode d'aide à la décision et d'industrialisation qui a pour objectif de simplifier l'intégration des processus liés à une gestion de projet incrémentale et itérative, agile. Elle s'applique de façon particulièrement efficace en association avec une méthode de développement telle que Scrum ou Lean. Il s'agit d'une méthode hybride, destinée à faciliter l'adoption de l'agilité au sein d'une organisation de taille significative. Tous les aspects de l'intégration de l'agilité dans le développement logiciel sont pris en charge en tenant compte du contexte global du projet et surtout de l'entreprise. Lorsqu'une méthode agile doit être appliquée à un projet impliquant des équipes importantes, DAD peut être d'un grand secours pour faciliter l'adoption des différents processus.

Entreprise Unified Process (EUP)

EUP est une extension des méthodes d'industrialisation comme DAD ou comme les dérivés de Unified Process (UP), ou des méthodes de développement agile comme Scrum. EUP apporte deux autres phases à la fin des cycles des autres méthodes agiles :

- Mise en production : maintenance en condition opérationnelle des systèmes déployés.
- Retrait de production : processus de retrait de la production des systèmes déployés.

La phase de retrait de production peut être mise en œuvre pour plusieurs raisons comme le remplacement complet du système, la fin du support de la version courante, la redondance du système ou encore l'obsolescence de l'application.

Chapitre 3 : RAD

3.1 Introduction

Le RAD signifie Développement Rapide d'Application.

3.2 Méthode de développement logiciel RAD

La **méthode RAD** et le **Processus Qualité RAD2**, impliquent 3 intervenants principaux (MOA, MOE, GAR[14]).

Pour la **maîtrise d'œuvre** les grandes phases de la démarche sont :

- Initialisation (mise en condition de l'organisation).
- Cadrage (expression des objectifs).
- Design (conception du futur système).
- Construction (développement de l'application).
- Finalisation (livraison des fonctionnalités attendues).

Le projet est piloté selon un suivi rigoureux des contraintes, des risques et de la qualité technique.

La **maîtrise d'ouvrage** doit assurer :

- L'expression des exigences et sa validation permanente.
- La préparation au changement organisationnel.
- La recette fonctionnelle et technique.
- Le démarrage.

Le projet est piloté selon un suivi rigoureux de la qualité fonctionnelle (rapport de Focus, suivis des divergences). Le **Groupe d'Animation et de Rapport** prend en charge les communications et la formalisation des informations.

3.2.1 Structure de développement

La structure méthodologique du projet s'appuie :

- la **méthode** RAD et son cycle semi-itératif2 en ce qui concerne les principes fondamentaux de conduite de projet ;
- le **processus** RAD2 pour l'ordonnancement pratique des opérations de développement ;
- une **techniques de modélisation** adaptée à la typologie de l'application (Merise/UML/Flux/ etc.).

Pour rappel (et en synthèse), la méthode RAD implique :

1. Un **cycle de développement** sécurisant et court fondé sur un phasage simple :
   Cadrage, Design, Construction et l'absolu respect d'une dimension temporelle (90 jours optimum, 120 jours maximum)
2. Une **architecture de communication** engageant des groupes de travail de structure et de composition variable selon les besoins des phases et respectant un mode opératoire précis structuré en trois étapes : pré-session, session, postsession.
3. Des **méthodes, techniques et outils** permettant de définir et d'appliquer des choix portant sur quatre natures d'objectifs potentiellement contradictoires :
   Budget, délais, qualité technique, qualité fonctionnelle et visibilité.
4. Une **architecture de conception** s'appuyant sur les techniques de l'objet et particulièrement sur celles qui permettent une conception « en vue de modifications ».

[14] Maîtrise d'ouvrage, maître d'œuvre, Groupe d'Animation et de Rapport

5. Une **architecture de réalisation** qui impose, pour garantir la qualité technique, des normes minimales, des revues de projet, des jalons zéro-défaut5 et qui recommande, pour garantir la qualité fonctionnelle, le prototypage actif et les Focus6 de visibilité.

3.2.2 Description globale des phases

La méthode RAD structure le cycle de vie du projet en 5 phases :

- L'**Initialisation** définit l'organisation, le périmètre et le plan de communication.
- Le **Cadrage** définit un espace d'objectifs, de solutions et de moyens.
- Le **Design** modélise la solution et valide sa cohérence systémique.
- La **Construction** réalise en prototypage actif (validation permanente).
- La **Finalisation** est un contrôle final de qualité en site pilote.

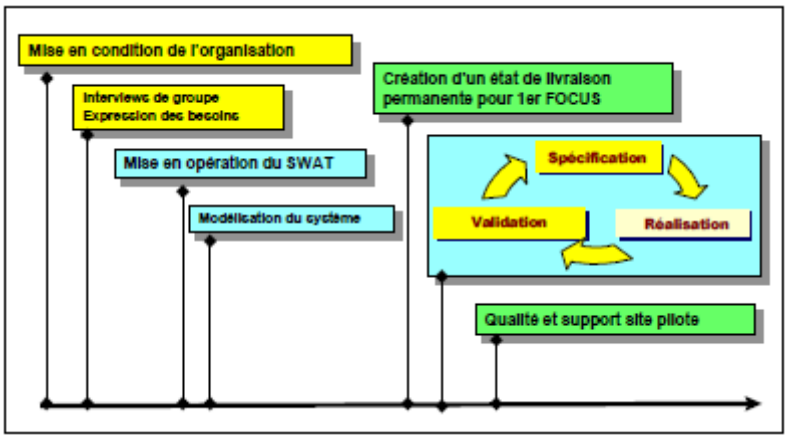


Figure 8 : Méthode RAD

Dans un second niveau de détail de ces phases comprennent :

1. **INITIALISATION** (préparation de l'organisation et communication)

Cette phase permet de définir le périmètre général du projet, de structurer le travail par thèmes, de sélectionner les acteurs pertinents et d'amorcer une dynamique de projet. Cette phase représente environ 6% du projet en charge.

2. **CADRAGE** (analyse et expression des exigences)

La spécification des exigences est du ressort des utilisateurs. Ils expriment leurs besoins lors d'entretiens de groupe. Il est généralement prévu de 2 à 5 jours de sessions par commission (thème). Cette phase représente environ 9% du projet.

3. **DESIGN** (conception et modélisation)

Les utilisateurs sont également impliqués dans cette étape. Ils participent à l'affinage et à la validation des modèles organisationnels : flux, traitements, données. Ils valident également le premier niveau de prototype présentant l'ergonomie et la cinématique générale de l'application. Il est prévu entre 4 et 8 jours de sessions par commission.
Cette phase représente environ 23% du projet. A partir de la phase de Design la parallélisassions du travail est possible (*figure 9*).

4. **CONSTRUCTION** (réalisation, prototypage)

Durant cette phase, l'équipe RAD (SWAT) doit construire l'application module par module. L'utilisateur participe toujours activement aux spécifications détaillées et à la validation des prototypes. Plusieurs sessions itératives sont nécessaires. Cette phase représente environ 50% du projet. A partir de la phase de Construction, à la parallélisassions du travail peut s'ajouter la sérialisation (figure 9).

5. **FINALISATION** (recette et déploiement)

Des recettes partielles ayant été obtenues à l'étape précédente, il s'agit dans cette phase d'officialiser une livraison globale et de transférer le système en exploitation et maintenance. Cette phase représente environ 12% du projet.

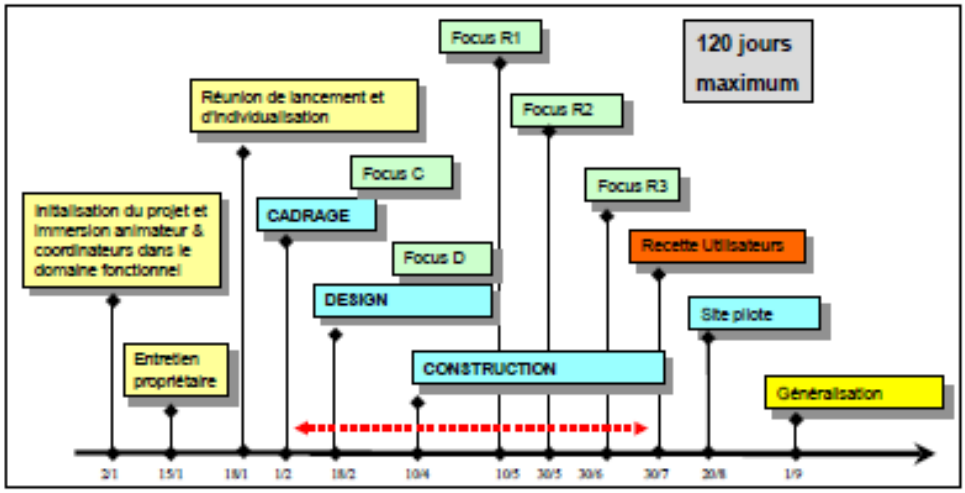


Figure 9:Principales étapes d'un cycle de projet à 120 jours

J'approfondis les phases précédentes :

- *Phase INITIALISATION* :

La phase d'**Initialisation** a pour objectif d'informer les intervenants des contraintes d'un projet RAD. Après une courte **Immersion** dans le domaine fonctionnel, le pilote, les coordinateurs et l'animateur présentent aux dirigeants et à la maîtrise d'ouvrage les contraintes de la méthode et le plan d'action à respecter. Un **plan de communication** est produit.

Cette étape a pour nom l'**Entretien propriétaire**. Si nécessaire, une opération de réingénierie des procédés « métier » précède l'automatisation du domaine.

Une **Réunion de lancement** du projet est ensuite organisée. Elle regroupe tous les acteurs recensés et donne lieu à l'individualisation des travaux préparatoires, à la **spécification des exigences** de la future application et à sa modélisation. Dans le cadre du plan de communication, les utilisateurs sont répartis dans des groupes de travail**8** organisés par thèmes. Cette étape doit s'effectuer en quelques dizaines de minutes.

■ *Phase CADRAGE* :

Lors du CADRAGE auquel participent les décideurs, l'animateur obtient le verrouillage des exigences, des budgets, des délais et de la solution globale sur les plans stratégique, fonctionnel, technologique et organisationnel.

Dans le cas où les exigences et les ressources divergeraient, il faut attribuer des priorités aux fonctionnalités en termes de retour sur investissement. Cette modélisation des traitements peut s'effectuer sous la forme d'une hiérarchie de fonctions et/ou suivant le principe des cas d'utilisation.

Le cadrage nécessite des équipes de taille intermédiaire9, incluant des utilisateurs de tous niveaux. Le processus est le suivant :

- dans chaque domaine, les thèmes principaux sont déterminés ;
- dans le cas d'un domaine stable, il n'est pas nécessaire de réaliser une étude détaillée de l'existant en session plénière ;
- dans les domaines où les fonctionnalités sont instables, un effort de réflexion devra être engagé et le cercle d'interlocuteurs élargi.

Modéliser en direct avec un vidéoprojecteur électronique. Dans le principe, la modélisation concrète des flux et des fonctionnalités devra être engagée le plus tôt possible avec un outil adéquat.

La durée des sessions de Cadrage est d'une demi-journée ou d'une journée. Il est possible en cas de contraintes de délais de réaliser des sessions pouvant atteindre 5 jours. Le Cadrage complet peut alors être réalisé en une seule session. Les sessions de Cadrage engagent simultanément les informaticiens de conception-développement (le SWAT) et les utilisateurs. Seules les validations d'un ensemble conséquent modélisé nécessitent un Focus.

■ *Phase DESIGN* :

La phase suivante, le DESIGN, repose sur la disponibilité d'un AGL de conception léger et puissant. Cet outil est utilisé « en direct », dans une salle spécialement équipée de moyens de vidéoprojection et de communication. Sous la coordination de l'animateur, les utilisateurs significatifs et les concepteurs-développeurs travaillent alors en commun et en direct à la modélisation détaillée des traitements et des données de l'application. La présentation d'un premier niveau de prototype conclut cette phase.

■ *Phase CONSTRUCTION* :

La phase de CONSTRUCTION affine le prototype. Elle fusionne les étapes classiques de spécification détaillée, de réalisation (codage), de tests unitaires et de tests d'intégration, ainsi que la plus grande partie des tests de cheminement fonctionnel qui constituent la recette initiale de l'application. Cette homogénéité constitue, avec la prise de décision immédiate et la validation permanente, les bases mêmes de la productivité et de la qualité RAD.

Pour atteindre cette performance, les outils de Construction utilisés doivent être choisis avec soin, une charte graphique doit avoir été validée, des normes de programmation employées, un modèle de transaction généralisé à tous les modules.

La validation permanente, comme son nom l'indique, est réellement permanente. Elle s'effectue à chaque séance de travail avec l'utilisateur.

La validation permanente garantit lors de chaque ajout de fonctionnalité la conformité au besoin. Le groupe de travail idéal comprend un membre du SWAT et un ou deux utilisateurs.
Un développement RAD se distingue par la mise en œuvre de plusieurs techniques de réalisation.

- **SWAT** : organisation d'une équipe de profil « concepteur-développeur » basée sur la compétence, la complémentarité, l'autonomie et la démocratie.
- **Normes de codage** : normes publiées et acceptées visant à uniformiser les techniques de codage.
- **Validation permanente** : engagement continu des utilisateurs dans le prototypage lors de réunions informelles qui engagent un ou deux utilisateurs par concepteur/développeur.
- **Revue de code** : principes planifiés et acceptés de vérifications croisées de la conformité des pratiques de codage aux normes publiées.
- **Jalons zéro-défaut (ZD) :** technique orientée « planning » ; elle permet de contrôler l'avancement de l'application en matière de visibilité et de qualité. Des jalons ZD peuvent être, si nécessaire, planifiés entre les Focus. Ils correspondent à une revue du code suivie d'une intégration et d'une validation complète de cheminement par l'utilisateur de base.
- **Focus** : réunion plénière de présentation et de validation du projet dans toutes ses dimensions.
- **Etat de livraison permanente** : une version de l'application est maintenue dans un état livrable suite à un Focus ayant confirmé le jalon ZD. Elle peut être présentée à tout moment ou utilisée si nécessaire en fonctionnalités réduites.
- **Techniques de conception en vue de modifications** : ont pour but de faciliter l'évolution ultérieure de l'application. Elles sont mises en œuvre lors de la phase de Design. En phase de Construction il existe des techniques dont le but est similaire.
  L'ensemble de ces techniques a pour finalité de permettre une évolution continue depuis la conception, la réalisation et jusqu'à la maintenance, prolongement naturel de la vie d'une application.
- **Structure de Construction itérative et incrémentale** : dès le début de la phase de Construction un plan d'évolution du prototype est précisé. Il définit et segmente l'ensemble des fonctionnalités à produire, lors de chaque borne de validation, appelée Focus**.** La segmentation peut être verticale ou horizontale :
  - une segmentation verticale repose sur un simple découpage en modules ou en écrans ;
  - une segmentation horizontale se base sur la notion de priorité des fonctions et impose souvent que l'ensemble des modules soit mis en chantier simultanément, mais à un niveau de fonctionnalité limité.

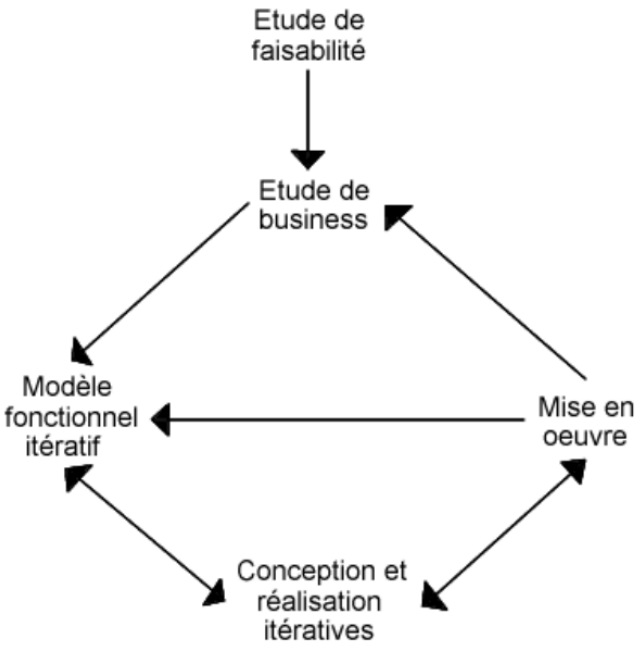
■ *Phase FINALISATION*:

Pour de multiples raisons dont notamment les techniques de validation permanente et de jalons zéro-défaut, la recette RAD est beaucoup moins conséquente qu'une recette classique.
La phase de FINALISATION comprend, entre autres, la formation, les activités liées au déploiement, le bilan de projet, le retour de connaissance. Elle clôture le projet RAD.

Chapitre 4 : DSDM

4.1 Présentation

Le DSDM (dynamic systems Development method) est un framework Agile très peu connu en France. Le DSDM est apparue en 1994 en Grande Bretagne pour dispenser au RAD d'une certaine discipline. Le DSDM (dynamic systems Development method) est un framework agile qui a évolué avec le temps ; il se concentre plutôt sur une approche de gestion de projet et de livraison que sur une façon de faire du code.



Le DSDM remis à jour en 2014

Il faut savoir que la dernière version du DSDM ne remonte qu'à 3 ans. Etant tout à fait agile, cette méthode n'hésite pas à se remettre au goût du jour.

D'ailleurs, le DSDM n'hésite pas à proposer aujourd'hui de la travailler avec d'autres approches comme :

- PRINCE2
- Managing Successful Programmes
- PMI-BOK
- Extreme Programming (une autre méthode agile).

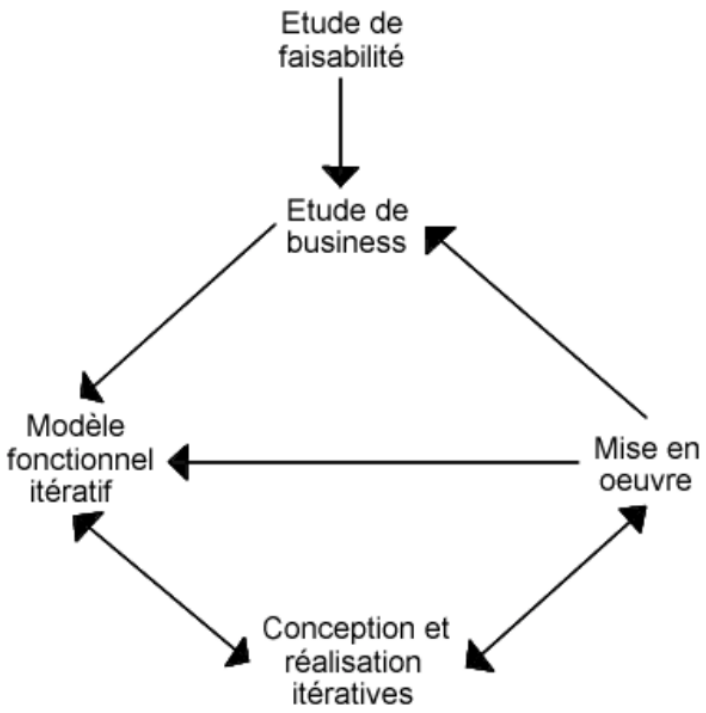
4.2 Les 9 principes

Voici la liste des 9 principes fondateurs du DSDM :

- **Implication des utilisateurs** -> ils doivent être présent tout au long du développement du projet
- **Autonomie** -> L'équipe est autonome sur le développement et doit avoir un droit de décision imparable sur l'évolution des besoins.
- **Visibilité du résultat** -> Livraison fréquente de l'application afin d'obtenir des feedback fréquents.
- **Adéquation** -> L'application livrée doit être en adéquation avec le besoin des utilisateurs.
- **Développement itératif et incrémental** -> Le produit doit constamment s'améliorer avec le feedback des utilisateurs.
- **Réversibilité** ->En cas de besoin, tout développement doit pouvoir revenir en arrière.
- **Synthèse** -> Un schéma directeur fixe le projet dans son ensemble que ce soit au niveau vision ou aux niveaux objectifs
- **Tests** -> Automatisation de tests fréquent afin de garantir une non régression au sein de l'application.
- **Coopération** -> L'ensemble de l'équipe ou parties prenantes sur le projet doivent accepter d'éventuelles modifications des fonctionnalités demandées.

4.3 Processus

Voici un schéma simple du processus du DSDM que nous allons expliquer plus concrètement à la suite :



Détails de chaque étape du dynamic systems Development method

Étude de faisabilité : déterminer s'il est opportun de faire le projet en question. -> Evaluation des coûts et de la valeur ajoutée attendue. Dans cette étape, on produit un *Rapport de Faisabilité* ainsi qu'un *Plan Global de Développement*. On développe parfois un prototype afin de démontrer la faisabilité technique.

Étude business (ou analyse fonctionnelle) : définition des spécifications -> fonctionnalités que l'application doit apporter, priorisation de celles-ci, dans un document appelé *Définition du Domaine Industriel*, mais aussi quels types d'utilisateurs sont concernés par l'application, de manière à pouvoir les impliquer. On définit également l'architecture du système, dans un document appelé *Définition de l'Architecture Système*. Enfin, à partir du *Plan Global de Développement*, on définit un *Plan Global de Prototypage*.

Modèle fonctionnel itératif et **Conception et réalisation itératives** : nous travaillons techniquement de façon itérative.

Mise en œuvre : mise en production de l'application afin de mettre notre application à disposition des utilisateurs types ; ceci permettra d'avoir des feedback rapides de ceux-ci et d'incrémenter le produit d'améliorations.

Chapitre 5 : FDD (Feature-driven development)

5.1 Présentation

FDD (Feature Driven-Development)

Le FDD (Feature-driven development) est une méthode agile créée par Jeff De Luca en 1997 ; cette méthode agile basée sur des concepts incrémental et itératif avait été créée pour une grande banque singapourienne.

Curieusement pour ceux qui découvrent cette méthode agile FDD beaucoup moins populaire qu'un Scrum, il faut savoir que sa première publication était au sein du livre Java Modeling in Color avec UML et non pas dans une oeuvre à part entière.

5.2 En quelques mots

Pour ceux qui sont dans les grands groupes, vous allez être étonné de découvrir cette méthodologie FDD car on y trouve beaucoup de choses qui se mettent en place dans ces grands groupes… Et ce sont mêmes des choses dites « non agile ».

En fait le FDD est ancien et il est vrai que l'agilité a en réalité évolué avec le temps. Cependant, on la considère comme l'une des méthodes agiles existantes malgré ses contradictions avec des méthodes agiles telles que XP ou Scrum.

Le FDD (Feature-driven development) propose :

- des itérations courtes et grandes
- d'avoir une meilleure communication durant l'ensemble du développement
- de faire des livraisons fréquentes avec de vrais travaux terminés

- des informations de progression et d'état précises et significatives pour un minimum de coût et de perturbation pour les développeurs.
- d'avoir des processus appréciés par les clients, développeurs et managers

5.3 Les itérations du FDD en 5 étapes

Une itération en FDD (Feature-driven development) se compose en 5 étapes différentes :

1. Créer le modèle du système avec un diagramme de classes UML
2. Faire la liste des fonctionnalités à réaliser
3. Assigner les fonctionnalités aux développeurs
4. Créer le modèle de chacune des fonctionnalités
5. Développer chacune des fonctionnalités

Contrairement au Scrum et à l'Extrême programming, le FDD recommande fortement d'assigner les fonctionnalités à un ou des développeurs précis.

5.4 Les 6 mots clés

La méthodologie FDD définit 6 rôles clés dont certains peu appréciés par la communauté agile d'aujourd'hui.

- **The Project Manager (PM)**

En FDD, il existe bien ce rôle tant rejeté par les communautés agiles d'aujourd'hui. Il aura comme rôle de faire les bons reportings d'avancement, de se battre pour le bien être de l'équipe, de s'occuper des budgets, du recrutement et de la logistique.

Si on regarde de près cette description, c'est souvent le rôle attribué aux Scrum Master dans les grands groupes qui ont du mal à devenir 100% Scrum Master.

- **The Chief Architect (CA)**

Le CA est responsable de la modélisation complète de l'architecture pour le produit en cours de développement.

- **The Development Manager (DM)**

Le DM est responsable de l'activité des développements. En effet, nous sommes dans une méthode agile totalement en opposition avec Scrum bien qu'au final, on retrouve plus ou moins ces types de rôles au sein des grands groupes.

Certains seront ravis de découvrir cette méthode agile pour dire qu'au final c'est agile de fonctionner avec des rôles aussi précis fortement déconseillés dans d'autres méthodes agiles… C'est pas totalement faux, disons que l'agilité à évolué et que les coachs préconisent dans une grande majorité la partage des responsabilités.

- **The Chief Programmers**

En FDD, les développeurs expérimentés travailleront sur la partie analyse et les spécifications techniques. On les rend responsables d'un groupe de 3 à 6 développeurs chacun.

- **The Class Owner**

Ce sont des développeurs qui travaillent en petits groupes justement sous le le Chief Programmer avec comme rôles de développer, de tester et de documenter les fonctionnalités réalisées.

- **The Domain Experts**

Ce rôle est en fait représenté par les Utilisateurs clés, Sponsors et Business Analyst. Ils sont présent pour identifier les besoins et permettent aux développeurs de travailler. Leur présence est indispensable pour avoir un produit de qualité.

Des rôles transverses en FDD

Il existe quelques rôles transverses proposés par la méthode comme le Release Manager, le Language Guru (formateur de développeurs), le Build Engineer (responsable des build applicatif), le Toolsmith (développeur de petits outils ou librairies pour les développeurs) et le System Administrator.

Le FDD (Feature-driven development) propose également de rajouter des testeurs, des Deployers et des Technical Writers (ceux qui écrivent les documents techniques).

5.5 Les pratiques du FDD

En FDD (Feature-driven development), il y a des pratiques qui sont très privilégiées. Nous devons par exemple avoir l'utilisation de l'UML Color (couleurs pour les classes) pour réaliser les modèles de classes.

Travaillons par feature

Le FDD privilégie fortement le développement par feature. Une « feature » doit comme une user-story être possible à développer en deux semaines. Si ce n'est pas le cas il faut découper la « feature » en deux features distinctes.

D'ailleurs, il existe un format classique pour écrire des features en FDD :

<action> the <result><by|for|of|to|><a(n)><object>

Par exemple, on pourrait avoir :

Calculer [action] le total [result] d'une vente [object].

En parlant de feature, les équipes en FDD sont définies en Feature Team qui seront responsables des parties sur lesquelles elles interviennent ; cela implique que la feature team sera choisi également à l'avenir pour des développements concernant leurs périmètres.

Ces feature teams seront d'ailleurs au maximum autonomes pour travailler sur leur périmètre.

Les développeurs sont responsables

Une pratique totalement en opposition avec les méthodes agiles aujourd'hui en place au sein des entreprises est le fait de faire des « Class Ownership ». En gros une classe dans le code est de la propriété d'un développeur (ou de deux développeurs) tant qu'il est au sein de l'entreprise. Si un développement doit se faire à nouveau sur cette classe à l'avenir, on privilégiera que son propriétaire travaille dessus.

On privilégie cependant quand cela est possible d'avoir deux Class Owner pour diminuer les craintes d'être seul responsable (ressenti du développeur) en cas de problème.

Des builds réguliers

Le FDD impose le fait d'avoir des build réguliers (ou mise en recette) afin de pouvoir tester régulièrement l'avancement du produit. La récupération de feedback est un point essentiel dans les produits agiles.

Outil de reporting d'avancement

Le FDD impose comme le fait d'ailleurs d'autres frameworks, le fait de faire du reporting d'avancement. Il est très important dans les projets d'avoir de la transparence sur l'avancement du projet afin de pouvoir gérer au plus vite les éventuels obstacles.

Chapitre 6 : Crytal

6.1 Présentation

Les méthodes agiles Crystal ont été créées au milieu des années 90 par Alistair Cockburn, un célèbre programmeur américain. Il a notamment participé à la rédaction du Manifeste Agile en 2001 et a publié plus d'une demi-douzaine de livres sur le développement agile entre 1997 et 2006.

Alistair Cockburn a étudié durant de nombreuses années la façon dont les différentes équipes de développement travaillaient. Il a mis en lumière le fait que bien que les équipes suivies ne respectaient pas à la lettre les méthodes de gestion de projet classiques, elles parvenaient tout de même à mener leurs projets à bien. Il a donc relevé, compilé et catalogué tout ce qui a permis à ces équipes de réussir leurs projets, afin de constituer les méthodes agiles de gestion de projet Crystal.

Les **méthodes agiles Crystal** sont généralement plus souples à mettre en œuvre que d'autres (elles ont été conçues dans ce sens) et fournissent un ensemble d'outils et de bonnes pratiques pour la gestion d'un projet.

Contrairement à d'autres, les méthodes agiles Crystal ne sont pas centrées sur les processus, qui passent au contraire au second plan. Partant du principe qu'une équipe de développement comprend des personnes aux compétences et aux talents très différents, le processus en lui-même n'est pas une priorité.

Les méthodes agiles Crystal sont donc plutôt centrées sur:

- les personnes,
- les interactions,
- la communauté,
- les compétences,
- les talents
- et la communication.

Plutôt que de définir des processus très précis et contraignants, les méthodes agiles Crystal mettent l'accent sur la communication et la collaboration entre les différents participants. C'est ce qui permet aux méthodes Crystal d'être plus « légères » à mettre en place et de s'adapter à la plupart des cas rencontrés dans les projets.

6.2 Famille de méthode agile crystal

Les différentes méthodes agiles Crystal ont été conçues pour s'adapter à la taille de l'équipe de développement. Un code de couleur est utilisé pour identifier le « poids » de la méthode agile à utiliser en fonction du projet.

Les différentes couleurs utilisées sont :

- transparent (« Clear »),
- jaune,
- orange,
- rouge,
- marron,
- diamant

- et saphir.

Ce qui avec une variante supplémentaire nous donne les méthodes Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Orange Web, Crystal Red, Crystal Maroon, Crystal Diamond et Crystal Sapphire. Plus le projet à gérer est important, et plus la couleur de la méthode est foncée.

On comprend donc que la méthode de management agile Crystal Clear est particulièrement adaptée à des petits projets alors que Crystal Yellow et Orange conviendront à des projets de taille moyenne. Pour des projets de très grande envergure ou particulièrement critiques, les méthodes Crystal Diamond et Crystal Sapphire seront choisies prioritairement.

Même si toutes les méthodes agiles Crystal présentent de nombreux points communs, il est particulièrement important de choisir la bonne couleur en fonction de la taille et de la criticité du projet afin de disposer des outils adaptés à la gestion de contraintes plus ou moins fortes.

6.3 Différentes méthode agile Crytsal mais 7 points communs

S'il existe différentes méthodes dans la famille Crystal adaptées à différents cas de figure et types de projets, toutes partagent sept points communs :

- des livraisons fréquentes,
- un processus d'amélioration continue,
- une communication accrue entre les membres de l'équipe,
- la mise en confiance de l'équipe,
- la priorité à la concentration des développeurs,
- l'accès facilité à des experts
- et un environnement technique disposant de tests automatisés et d'outils permettant d'effectuer de la gestion de configuration et des intégrations fréquentes.

1- Livraisons fréquentes

Les livraisons fréquentes font parties des pratiques de la plupart des méthodes agiles. L'équipe de développement sélectionne les fonctionnalités incluses dans chaque livraison et crée les tests nécessaires. Avec les méthodes agiles Crystal, les livraisons peuvent être plus ou moins rapprochées, toutes les semaines, ou tous les trimestres, en fonction de la taille et de la longueur du projet.

Grâce à des livraisons fréquentes, les membres de l'équipe sont en mesure de détecter plus rapidement les anomalies. Plus une anomalie est détectée tôt dans un projet, et moins elle coûte cher à corriger. De même, si le client s'aperçoit finalement que l'application développée ne correspond pas exactement à ce qu'il voulait, les changements pourront être pris en compte plus facilement que s'il fallait attendre la fin du projet. Contrairement à d'autres méthodes agiles, il est possible qu'une livraison soit composée de plusieurs itérations de développement.

2- Processus d'amélioration continue

Le processus d'amélioration continue a pour objectif de permettre à l'équipe de développement de quitter temporairement ses tâches habituelles pour se pencher sur les problèmes qui ont pu être rencontrés. L'objectif est d'analyser les problèmes, d'identifier les causes et de mettre en place les solutions pour qu'ils ne se reproduisent plus, de tirer les leçons de l'expérience acquise. C'est

l'occasion également d'optimiser les processus du projet afin que l'équipe soit encore plus efficace d'une itération à l'autre.

Dans les méthodes agiles Crystal, des réunions de réflexion sur l'amélioration continue du projet sont organisées toutes les deux semaines. C'est l'occasion pour l'équipe projet de faire le tour de ce qui a bien fonctionné et de ce qui a moins bien fonctionné afin d'identifier les axes d'amélioration.

3- Communication entre les membres de l'équipe

La communication entre les membres de l'équipe projet est vivement encouragée. Pour ce faire, l'équipe entière est de préférence regroupée dans une pièce unique. Si l'équipe est trop importante, au-delà de huit personnes, des cloisonnements peuvent être mis en place de façon à maintenir la communication, tout en limitant les sources de distraction et les nuisances. Le but du regroupement de l'équipe en un seul lieu est de limiter les sources de distraction des développeurs. S'ils ont une question à poser ou une demande à faire, ils n'ont pas à se déplacer, à changer de pièce, ils rompent moins longtemps leur concentration et sont plus efficaces.

Les informations circulent plus facilement entre les membres de l'équipe, les questions trouvent des réponses plus rapidement. De plus, en étant proches les uns des autres, les développeurs assistent à toutes les conversations et peuvent ainsi partager leurs expériences et trouver de nouvelles idées. Ce type d'organisation permet également de libérer la créativité de l'équipe projet.

4- Mise en confiance de l'équipe

Les méthodes agiles Crystal mettent en avant la confiance entre les membres de l'équipe projet. Aucune suggestion, aucune idée ne doit être écartée, tout le monde peut s'exprimer sans avoir la crainte de se voir ridiculisé. Chacun doit avoir suffisamment confiance dans les autres membres de l'équipe pour se sentir libre d'exposer ses idées et ses objections. La confiance est un élément central des méthodes agiles Crystal.

5- Concentration de l'équipe projet

La concentration dans les méthodes agiles Crystal concerne deux domaines :

- les tâches attribuées individuellement
- et la direction suivie par le projet dans son ensemble.

Afin de permettre aux membres de l'équipe de se concentrer sur les tâches qui leur ont été attribuées, toutes les sources de distractions doivent être proscrites, ou plus raisonnablement limitées le plus possible : interruptions pour des discussions, réunions, questions posées hors contexte, conversations téléphoniques…

Toute interruption de la concentration d'une personne sur sa tâche en cours peut être grandement préjudiciable, car elle aura besoin d'un certain temps pour revenir à son niveau de concentration initial. Il y a donc une perte de temps et de productivité évidente. Pour éviter cela, l'équipe projet devra être mise à l'abri autant que faire se peut de toutes ces distractions.

Ainsi, les méthodes Crystal définissent deux règles simples :

- un développeur doit travailler sans interruption durant une période de deux heures (on s'isole, on coupe le téléphone et tout moyen de communication susceptible d'interrompre la tâche en cours)
- et un développeur ne doit pas travailler moins de deux jours complets sur un projet avant d'en changer si nécessaire (en-dessous de deux jours, il n'a pas le temps de s'approprier le projet et d'atteindre son efficacité maximale).

Enfin, la direction suivie par le projet, ses objectifs, les délais, toutes ces informations doivent être parfaitement claires et connues de l'ensemble de l'équipe projet, de façon à se concentrer sur les tâches prioritaires.

6- Accès facilité à des experts

Le ou les experts techniques et métier associés au projet doivent se rendre disponibles pour répondre à toute question pouvant se poser l'équipe de développement. Ils doivent être totalement impliqués dans le projet, et facilement accessibles. Au moindre doute énoncé par un développeur concernant telle ou telle fonctionnalité, l'expert doit être en mesure de lui répondre dans les plus brefs délais, et de façon la plus précise possible. Il est courant que l'expert ne soit pas disponible 100% du temps pour le projet. Dans ce cas, il devra être possible d'organiser avec lui au moins une réunion hebdomadaire de deux heures, et de le joindre par téléphone à n'importe quel moment.

7- Environnement technique

Enfin, le dernier point commun entre toutes les méthodes agiles Crystal concerne l'environnement technique utilisé par l'équipe de développement. Tous les outils nécessaires à l'intégration continue des développements (tests et déploiements automatisés, remontées d'alertes en cas de détection d'anomalie…) doivent être mis en place dès le début du projet. C'est à cette condition que les anomalies pourront être détectées rapidement, et corrigées.

Plus une anomalie est détectée tôt, et moins le coût de sa correction sera élevé. L'utilisation de ces outils et d'un gestionnaire de code source performant (SVN et Git sont les plus répandus actuellement) permettra de détecter les anomalies mais également de revenir à une version précédente du code source si les modifications nécessaires sont trop lourdes pour être faites rapidement. Un outil de gestion des configurations va également permettre d'effectuer des livraisons avec des paramétrages différents, adaptés à l'environnement ciblé (la configuration sera différente suivant que la livraison à lieu sur un serveur de tests, d'intégration, de recette ou de production).

Chapitre 7 : eXtreme Programming

7.1 Présentation

La méthodologie eXtreme Programming ou XP est une méthode de gestion de projet qui applique à l'extrême les principes du développement agile, c'est-à-dire se concentrer sur les besoins du clients, mettre en place un développement itératif et l'intégration continue. L'équipe projet et ses relations avec le client sont au coeur de XP.

Cette méthode a été créée par Kent Beck entre 1996 et 1999, lorsqu'il travaillait sur un projet pour Chrysler. Elaborée à l'origine pour le secteur informatique, eXtreme Programming est aujourd'hui très populaire car **elle fonctionne pour tous types de projets, de toutes tailles et de tous secteurs confondus, partout dans le monde**. Cette méthodologie est idéale pour de petites et moyennes équipes, c'est-à-dire pas plus d'une vingtaine de personnes environ.

7.2 Principe

Les principes de la méthode eXtreme Programming ne sont pas nouveaux puisqu'il s'agit de ceux des méthodes Agiles. La différence et l'originalité résident dans le fait qu'ils sont poussés à l'extrême.

La méthode eXtreme Programming s'appuie sur :

- une forte réactivité au changement des besoins du client ;
- un travail d'équipe ;
- la qualité du travail fourni ;
- la qualité des tests effectués au plus tôt.

XP repose sur cinq valeurs fondamentales :

- **Communication** : il est essentiel que chaque membre de l'équipe communique quotidiennement avec ses collègues ainsi qu'avec le client. C'est un moyen incontournable pour résoudre les problèmes.
- **Simplicité** : la façon la plus simple d'arriver au résultat est privilégiée. L'équipe projet fait ce qui est nécessaire et demandé, rien de plus. Une application simple sera plus facile à faire évoluer ensuite.
- **Feedback** : le retour d'information entre l'équipe projet et le client est essentiel. Chaque étape du projet est envoyée aussi rapidement et souvent que possible au client afin qu'il teste, donne son avis et valide l'étape. Chaque demande de modification est prise en compte immédiatement.
- **Respect** : le respect de chaque membre de l'équipe et de son travail sont primordiaux. Le management, l'équipe projet et le client se respectent mutuellement.
- **Courage** : Il faut du courage pour effectuer certains changements comme essayer une nouvelle technique, recommencer une itération non validée ou revoir l'organisation du projet. Le courage permet de sortir d'une situation inadaptée.

7.3 Fonctionnement

Les cinq valeurs de XP se déclinent en **treize pratiques** qui se renforcent mutuellement :

- **Client sur site** : le client doit être représenté sur place pendant toute la durée du projet. Ce représentant doit avoir une vision globale du résultat à obtenir et être disponible pour répondre aux questions de l'équipe.
- **Jeu du planning (ou planning poker)**: le planning est réalisé en collaboration avec le client. Ce dernier crée des scénarios pour les fonctionnalités qu'il souhaite obtenir. L'équipe évalue le temps nécessaire pour les mettre en œuvre. Le client sélectionne ensuite les scénarios en fonction des priorités et du temps disponible.
- **Intégration continue** : lorsqu'une tâche est terminée, elle est tout de suite intégrée dans le produit complet. Cela permet d'éviter la surcharge de travail due à l'intégration de tous les éléments avant la livraison. Les tests facilitent cette intégration : quand tous les tests sont positifs, l'itération est terminée.
- **Petites livraisons**: les livraisons doivent être les plus fréquentes possibles afin que le client donne son avis et que les modifications soient rapidement prises en compte par l'équipe.
- **Rythme soutenable** : aucune heure supplémentaire n'est tolérée. S'il y en a, alors le planning doit être revu. Un collaborateur fatigué travaille mal et fait plus d'erreurs.
- **Tests fonctionnels** : à partir des scénarios définis par le client, l'équipe crée des procédures de test qui permettent de vérifier l'avancement du développement. Lorsque tous les tests fonctionnels passent, l'itération est terminée.
- **Tests unitaires** : pour chaque fonctionnalité, un test est écrit afin de vérifier qu'elle fonctionnera comme prévu. Ce test sera conservé jusqu'à la fin du projet, tant que la fonctionnalité est requise. À chaque modification du code, tous les tests sont lancés afin d'identifier immédiatement s'il y a un problème de fonctionnement.
- **Conception simple** : on va droit à l'essentiel en se focalisant uniquement sur les besoins actuels des clients. Plus l'application est simple, plus il sera facile de la faire évoluer lors des prochaines itérations.
- **Utilisation de métaphores**: les équipes XP utilisent des métaphores pour décrire le système et son fonctionnement afin de clarifier les fonctionnalités à atteindre. Tout le monde parle le même langage.
- ***Refactoring* (ou remaniement du projet)**: le projet est amélioré régulièrement. Le but étant d'avoir de bonnes bases et de meilleures conditions de travail pour l'équipe.
- **Appropriation collective du projet** : la responsabilité du projet est collective. Chaque membre de l'équipe peut modifier toutes les portions du projet, mêmes celles sur lesquelles il n'a pas travaillé. L'objectif est d'être efficace et rapide.
- **Standards de langage** : puisque tout le monde travaille ensemble sur le projet, il est essentiel de faciliter le travail de chacun en utilisant les mêmes termes, le même style et des règles de communication claires.
- **Travail en binôme** : les collaborateurs travaillent en binôme. Le pilote et le copilote changent régulièrement afin d'améliorer la communication et la connaissance collective du projet.

Les projets gérés par la méthode eXtreme Programming reposent sur des cycles de développement (itérations) courts et rapides qui sont réalisés collectivement par l'équipe projet et le client dont l'implication est constante.

Les activités contre-productives ont été supprimées afin de réduire les coûts et la frustration de toutes les personnes impliquées.

7.4 Conditions de réussite

Pour être mise en place de façon efficace, la méthodologie eXtreme Programming nécessite un changement de mentalité et une acceptation par l'ensemble des équipes impliquées.

Afin de garantir le succès de XP, **il est indispensable que le client ou un représentant soit totalement investi dans le projet.**

Cette méthode ne peut pas fonctionner avec des grandes équipes, si le travail en binôme est impossible ou encore si les feedback sont longs et difficiles à obtenir.

Chapitre 8 : Devops

8.1 Qu'est-ce que le DevOps ?

Philosophie en constante évolution, le DevOps offre un framework conçu pour dynamiser et améliorer le développement d'applications et accélérer la mise à disposition de nouvelles fonctionnalités, de mises à jour logicielles ou de produits.

Il favorise la communication, la collaboration, l'intégration, la visibilité et la transparence continues entre les équipes chargées du développement d'applications (Dev) et celles responsables des opérations IT (Ops).

Cette relation plus étroite entre Dev et Ops se reflète dans chaque phase du cycle de vie DevOps : planification logicielle initiale, codage, développement, test, publication, déploiement, opérations et surveillance continue. Elle génère de façon constante des retours clients, ce qui renforce le potentiel d'amélioration lors du développement, des tests et du déploiement. La publication accélérée et permanente des modifications ou ajouts de fonctionnalités en est un exemple.

Les objectifs du DevOps s'articulent autour de quatre catégories : culture, automatisation, mesure et partage. Dans chacun de ces domaines, les outils DevOps améliorent la rationalisation et la collaboration des workflows de développement et d'opérations en automatisant les tâches chronophages, manuelles ou statiques des phases d'intégration, de développement, de test, de déploiement ou de surveillance.

8.2 Pourquoi le DevOps est-il important ?

En favorisant la communication et la collaboration entre les équipes chargées du développement et des opérations IT, le DevOps vise à optimiser la satisfaction client et à proposer des solutions à valeur ajoutée plus rapidement. Le DevOps est aussi conçu pour stimuler l'innovation dans une optique d'amélioration continue des processus.

Les pratiques DevOps accélèrent, optimisent et sécurisent la valeur commerciale des entreprises, par exemple via la publication plus fréquente ou la mise à disposition plus rapide de versions, de fonctionnalités ou de mises à jour des produits, le tout en assurant les niveaux de qualité et de sécurité appropriés. Autre objectif : améliorer les délais de détection, de résolution de bogues ou d'autres problèmes et de republication d'une version.

L'infrastructure sous-jacente apporte également au DevOps la fluidité des performances, la disponibilité et la fiabilité requises lors des étapes de développement, de test et de mise en production des logiciels.

8.3 Méthode DevOps

Afin d'accélérer et d'améliorer le développement et le lancement de leurs produits, les entreprises disposent de plusieurs méthodologies et pratiques DevOps de développement logiciel. Les méthodes Scrum, Kanban et Agile sont les plus couramment utilisées.

- **Scrum.** La méthode Scrum définit la manière dont les membres de l'équipe doivent collaborer pour accélérer les projets de développement et d'assurance qualité. Les pratiques

Scrum utilisent des workflows clés, une terminologie spécifique (sprint, time box, daily scrum) et des rôles désignés (Scrum Master, product owner ou responsable de produit).

- **Kanban.** Développée par Toyota pour améliorer l'efficacité de ses usines de montage, la méthode Kanban repose sur un suivi des travaux en cours (TEC) dans un projet logiciel à l'aide d'un tableau de Kanban.
- **Agile.** Les premières méthodes de développement logiciel agile continuent d'influencer largement les pratiques et les outils DevOps. De nombreuses approches DevOps, notamment Scrum et Kanban, intègrent des éléments de la programmation agile. Certaines pratiques agiles offrent une meilleure réactivité face à l'évolution des besoins en documentant les exigences sous forme de user stories, en organisant des réunions quotidiennes (daily standups) et en intégrant les retours des clients de manière continue. La méthodologie agile préconise également des cycles de développement logiciel plus courts, à l'encontre des méthodes classiques dites « en cascade » plus chronophages.

8.4 Chaîne d'outils DevOps

Les adeptes des pratiques DevOps intègrent souvent des outils compatibles DevOps dans leur « chaîne d'outils » pour rationaliser, accélérer et automatiser davantage les différentes étapes du workflow (ou « pipeline ») de fourniture des logiciels. Ces outils renforcent les principes fondamentaux du DevOps tels que l'automatisation, la collaboration et l'intégration entre les équipes chargées du développement et des opérations. Voici quelques exemples d'outils employés à différentes étapes du cycle de vie DevOps.

- **Planification.** Cette phase permet de définir la valeur commerciale et les exigences. Jira et Git peuvent être utilisés pour le suivi des problèmes connus et la gestion des projets.
- **Code.** Cette phase inclut la conception logicielle et la création du code logiciel à l'aide des logiciels GitHub, GitLab, Bitbucket ou Stash, par exemple.
- **Création.** Cette phase consiste à gérer les versions logicielles et à exploiter des outils automatisés pour compiler et intégrer le code en vue de sa mise en production. Des référentiels de code source ou de package « empaquettent » aussi l'infrastructure requise pour la livraison du produit à l'aide des logiciels Docker, Ansible, Puppet, Chef, Gradle, Maven ou JFrog Artifactory, par exemple.
- **Test.** Cette phase comprend des tests continus, qu'ils soient manuels ou automatisés, et vise à assurer une qualité de code optimale à l'aide des logiciels JUnit, Codeception, Selenium, Vagrant, TestNG ou BlazeMeter, par exemple.
- **Déploiement.** Cette phase peut inclure des outils de gestion, de coordination, de planification et d'automatisation de la mise en production des produits, avec Puppet, Chef, Ansible, Jenkins, Kubernetes, OpenShift, OpenStack, Docker ou Jira, par exemple.
- **Exploitation.** Cette phase permet de gérer les logiciels en production à l'aide des logiciels Ansible, Puppet, PowerShell, Chef, Salt ou Otter, par exemple.
- **Supervision.** Cette phase permet d'identifier les problèmes affectant une version logicielle en production et de collecter les informations correspondantes à l'aide des logiciels New Relic, Datadog, Grafana, Wireshark, Splunk, Nagios ou Slack, par exemple.

8.5 Pratiques DevOps

Les pratiques DevOps améliorent en continu et automatisent les processus. Bon nombre d'entre elles portent sur une ou plusieurs phases du cycle de développement :

- **Développement continu.** Cette pratique couvre les phases de planification et de codage dans le cycle de vie DevOps et peut inclure des mécanismes de contrôle des versions.

- **Tests continus.** Cette pratique prévoit des tests automatisés, planifiés et continus lors de l'écriture ou de la mise à jour du code de l'application qui accélèrent la livraison du code en production.
- **Intégration continue.** Cette pratique rassemble des outils de gestion de la configuration, de test et de développement pour assurer le suivi de la mise en production des différentes portions du code. Elle implique une collaboration étroite entre les équipes responsables des tests et du développement pour identifier et résoudre rapidement les problèmes de code.
- **Livraison continue.** Cette pratique automatise la publication des modifications du code après la phase de test, dans un environnement intermédiaire ou de préproduction. Un membre de l'équipe peut décider de publier ces modifications dans l'environnement de production.
- **Déploiement continu.** À l'instar de la livraison continue, cette pratique automatise la publication d'un code nouveau ou modifié dans l'environnement de production. Les entreprises peuvent être amenées à publier plusieurs fois par jour des modifications du code ou des fonctionnalités. Dans un contexte de déploiement continu, les technologies de conteneur comme Docker et Kubernetes assurent la cohérence du code entre plusieurs plateformes et environnements.
- **Surveillance continue.** Cette pratique prévoit une surveillance continue du code exécuté et de l'infrastructure sous-jacente. Les développeurs reçoivent des retours sur les bogues ou sur les problèmes.
- **Infrastructure-as-code.** Cette pratique peut être suivie dans plusieurs phases DevOps pour automatiser le provisionnement de l'infrastructure requise pour une version logicielle. Les développeurs ajoutent le « code » de l'infrastructure à l'aide de leurs outils de développement. Par exemple, un développeur peut créer un volume de stockage à la demande via Docker, Kubernetes ou OpenShift. Cette pratique permet aussi aux équipes chargées des opérations de surveiller les configurations de l'environnement, d'effectuer le suivi des modifications et de simplifier leur restauration.

8.6 Avantages DevOps

Pour les entreprises qui suivent les pratiques DevOps, les avantages commerciaux et techniques sont évidents et la plupart contribuent à améliorer la satisfaction des clients :

- Accélération et amélioration de la fourniture des produits
- Résolution plus rapide des problèmes et complexité réduite
- Plus grande évolutivité et disponibilité inégalée
- Stabilité accrue des environnements d'exploitation
- Meilleure utilisation des ressources
- Automatisation accrue
- Meilleure visibilité sur les résultats du système
- Innovation renforcée

Chapitre 9 : Méthode Prince 2

9.1 Introduction

PRINCE2 signifie PRojects IN Controlled Environments, et le chiffre “2“ fait référence à la seconde version de la méthode, publiée en 1996. Il s'agit d'une méthode pragmatique, structurée, évolutive et adaptable qui permet d'organiser, de gérer et de contrôler efficacement tous types de projets, quelle que soit sa taille. Cette méthode est aujourd'hui très répandue à l'international.

9.2 Historique

La méthode PRINCE2 a été créée au Royaume-Uni, mais ses atouts sont aujourd'hui reconnus dans le monde entier.

En 1975, la société britannique Simpact Systems Ltd crée une méthode de gestion de projet appelée PROMPT (Project Reporting, Organization & Management Planning Technique), qui se base sur des expériences pratiques réussies.

En 1979, le Central Computer and Telecommunications Agency (CCTA), qui deviendra l’Office of Government Commerce (OGC) en 1980, définit la méthode PROMPT comme la norme à utiliser pour tous les projets informatiques du Royaume-Uni.

En 1989, la méthode bénéficie d'une refonte importante et devient PRINCE.

En 1996, PRINCE s'étend et devient plus flexible afin de pouvoir gérer tous types de projets, de toutes envergures. La méthode est rebaptisée PRINCE2. Elle sera actualisée à deux reprises en 2005 et 2009, mais conservera sa dénomination.

9.3 Principe

La méthode PRINCE2 fournit un cadre et des directives permettant d'entreprendre la gestion de projet avec efficacité.

PRINCE2 se base sur :

- 7 principes qui présentent les lignes directrices à suivre ;
- 7 thèmes qui décrivent les aspects de la gestion de projet à aborder en permanence pour mener à bien le projet ;
- 7 processus qui détaillent les activités à accomplir pour réaliser le projet.

Les 7 principes

La méthode PRINCE2 s'appuie sur sept principes fondamentaux dont découle toute son organisation :

- Justification permanente de la raison d'être du projet ;

- Capitalisation, c'est-à-dire tirer les enseignements de l'expérience acquise lors de précédents projets ;
- Définition des rôles et des responsabilités de chaque membre de l'équipe ;
- Découpage du projet en étapes ;
- Gestion par exception, ce qui implique des responsabilités bien définies pour chaque niveau hiérarchique ;
- Concentration sur le projet, c'est-à-dire sur la définition, la livraison et la qualité attendues ;
- Adaptabilité de la méthode à l'environnement du projet, mais également à sa taille, sa complexité, son importance et ses risques potentiels.

Les 7 thèmes

Ces thèmes décrivent les aspects du management de projet qui doivent être abordés en permanence tout au long du projet. Ils sont au nombre de sept, comme les principes dont ils s'inspirent.

- **Le cas d'affaire** : il permet de répondre à la question « pourquoi faites-vous ce projet ? ». Il s'agit de la justification de l'existence du projet pour l'entreprise. Sa réalisation doit créer de la valeur pour l'entreprise. Si un projet ne dispose pas d'un cas d'affaire valide, il ne doit pas être lancé et si la justification du projet est remise en cause durant sa réalisation, celui-ci doit être arrêté.
- **L'organisation** : il s'agit de définir les rôles et les responsabilité des membres de l'équipe. Chacun doit savoir précisément ce qu'il a à faire afin de gérer efficacement le projet.
- **La gestion de la qualité** : le projet doit respecter les exigences du client en matière de qualité.
- **La planification** : il s'agit de définir les étapes nécessaires à la réalisation du projet.
- **La gestion des risques** : ce thème décrit la mise en place d'une procédure efficace de gestion des risques afin d'identifier, d'évaluer et de maîtriser les risques potentiels liés au projet.
- **La gestion des changements** : ce thème explique comment gérer les différents changements qui peuvent survenir lors de la réalisation du projet (problèmes de fournisseurs, modifications du client, etc.).
- **Le contrôle de la progression** : il explique comment surveiller et contrôler la progression du projet, et vérifie si le projet conserve sa raison d'être.

La méthode PRINCE2 exige la mise en pratique de tous les thèmes, et ce tout au long du projet.

Les 7 processus

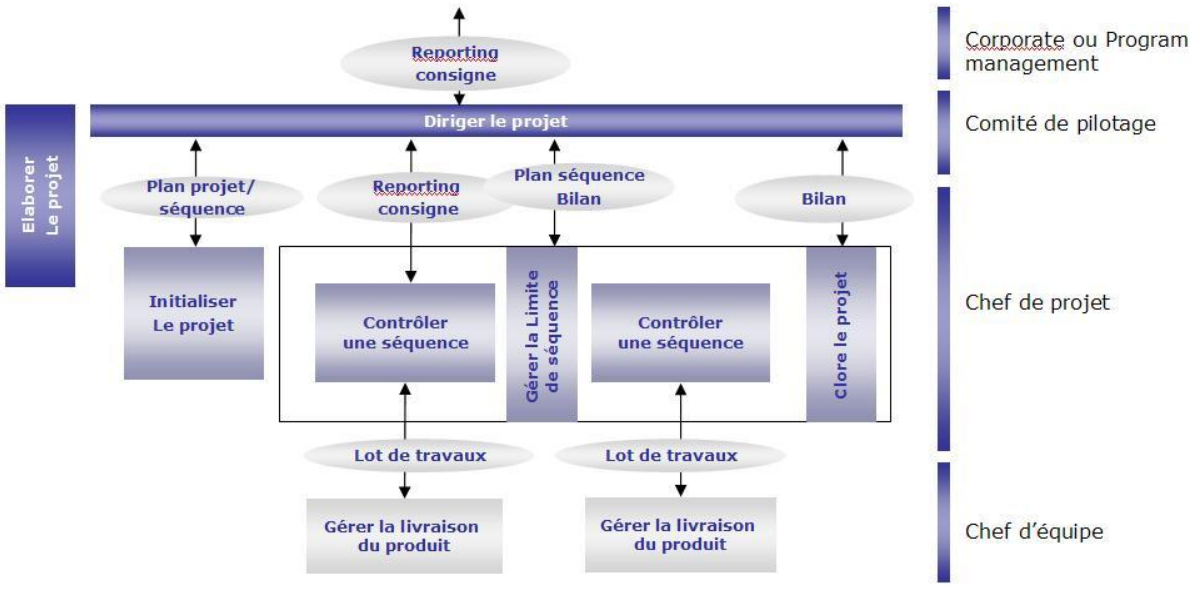
Un processus est un ensemble structuré d'activités conçues pour réaliser un objectif spécifique. Les processus décrivent les étapes de la vie du projet, du lancement jusqu'à la clôture. Chaque processus fournit une liste de contrôle des activités que chaque membre de l'équipe doit exécuter afin d'obtenir un résultat précis.

PRINCE2 compte sept processus qui regroupent l'ensemble des activités nécessaires pour organiser, gérer et livrer un projet avec succès :

- **Elaborer le projet** : l'objectif de ce processus est d'autoriser ou non le démarrage du projet.

- **Diriger le projet** : le but est de permettre au comité de pilotage de suivre et contrôler le projet pendant toute sa durée (le management quotidien du projet est délégué au chef de projet).
- **Initialiser le projet** : il s'agit d'un processus clé qui produit la documentation d'initialisation du projet permettant de définir précisément le périmètre et le contexte du projet (stratégies de communication, risques, configuration et qualité, le cas d'affaires détaillé et les différents contrôles du projet).
- **Contrôler une séquence** : il s'agit du travail quotidien du chef de projet. L'objectif est de mesurer et contrôler le déroulement et l'exécution de chaque phase.
- **Gérer la livraison du projet** : le but de ce processus est de gérer la relation entre les équipes projet et le chef de projet en établissant des exigences formelles pour l'acceptation, l'exécution et la livraison des travaux du projet. Les équipes informent le chef de projet de la progression du projet. Le but est de garantir que le projet livré respecte les exigences en matière de qualité.
- **Gérer une limite de séquence** : il s'agit de fournir au comité de pilotage les éléments permettant de valider la réussite et donc la fin de la phase en cours, et ainsi d'autoriser le démarrage de la phase suivante.
- **Clôturer le projet** : l'objectif est de clore proprement le projet, de l'évaluer et d'établir le bilan de fin de projet. Ce processus permet également d'archiver les expériences et les observations vécues pour de futurs projets.

J'explique plus en détails les 7 processus :



Elaborer le projet (SU - Starting Up a projet) dont l'objectif est d'autoriser le démarrage du projet. Ce processus doit permettre notamment de clarifier :

- Quelle est la mission du projet

- Qui doit intervenir dans l'équipe projet
- Quelle approche utiliser pour le projet
- Quelles sont les attentes du client, en termes de qualité
- Quels sont les risques identifiés
- Quelles sont les dates clés (établir une première planification)

Ce processus est en général extrêmement court.

Diriger le projet (DP - Directing a project) dont l'objectif est de permettre au comité de pilotage de suivre et contrôler le projet, à travers le suivi de rapports d'avancement, les points de validation et une gestion par exception (la gestion courante étant assurée par le chef de projet). La direction de projet se focalise sur les points suivants

- La bonne mise en place du projet
- La gestion du périmètre de chaque étape (pour éventuellement identifier des besoins de ressources supplémentaires)
- La direction de projet (suivi de l'avancement, fourniture de conseils et d'orientations, réaction aux principales menaces et opportunités identifiées)
- La fin de projet (pour contrôler la bonne fin du projet)

Initialiser le projet (IP - Initiating a project) dont l'objectif est de définir de façon plus précise le périmètre du projet (conformément aux éléments précisés dans le processus de démarrage du projet)

- Définir comment obtenir la qualité attendue des produits à fournir
- Planifier et chiffrer le projet
- Documenter la justification métier (business case) à l'origine du projet (et confirmer que le projet correspond à un besoin réel)
- S'assurer que la durée et l'effort requis pour le projet sont justifiés, en prenant en compte les risques identifiés
- Permettre et encourager la direction de projet à assure son rôle de management
- Définir la ligne de conduite pour la prise de décision durant le projet
- Accepter les ressources pour la suite du projet (ou demander un ajustement)

Le livrable principal de ce processus est le document d'initialisation projet (PID : Project Initiation Document), qui définit le périmètre et le contexte projet.

Ce processus initialise également les comptes rendus qualité, le journal des questions soulevées et la base des leçons tirées du projet, documents qui seront complétés tout au long du projet.

Enfin, la mise en place du projet étant une phase comme une autre, elle se termine par la livraison du plan projet pour la phase suivante.

Contrôler une séquence (CS - Controlling a stage) dont l'objectif est de mesurer et contrôler la bonne exécution de la phase. Ce processus constitue le travail quotidien du chef de projet, avec

- Autoriser le lancement des taches à effectuer
- Récolter les informations sur l'état d'avancement des taches en cours

- Surveiller les écarts
- Analyser la situation de la phase
- Etablir les rapports d'avancement
- Prendre les actions correctives nécessaires

Gérer la limite de séquence (SB - Managing stage boundaries) dont l'objectif est de fournir au comité de pilotage les éléments permettant de valider la bonne fin de la phase courante

- Assurer la direction de projet que l'ensemble des livrables de la phase ont été correctement délivrés
- Fournir les informations nécessaires à la direction de projet sur la visibilité du projet
- Fournir toutes les autres informations nécessaires à la direction de projet pour qu'elle confirme le passage à la phase suivante
- Capitaliser l'expérience acquise dans la phase terminée, et documenter les leçons apprises

Les documents produits lors de ce processus doivent permettre de valider la bonne fin de la phase, de planifier la phase suivante et de comparer le déroulement effectif par rapport au déroulement prévu pour en tirer des enseignements pour la suite.

Les documents suivants sont normalement publiés à l'issue de ce processus :

- Un rapport de fin de phase, fournissant les informations nécessaires à la direction de projet pour clore cette phase
- Le plan projet effectif de la phase terminée, avec les indicateurs de performance à comparer avec le plan projet initial
- Le plan projet de la phase suivante à valider par la direction de projet

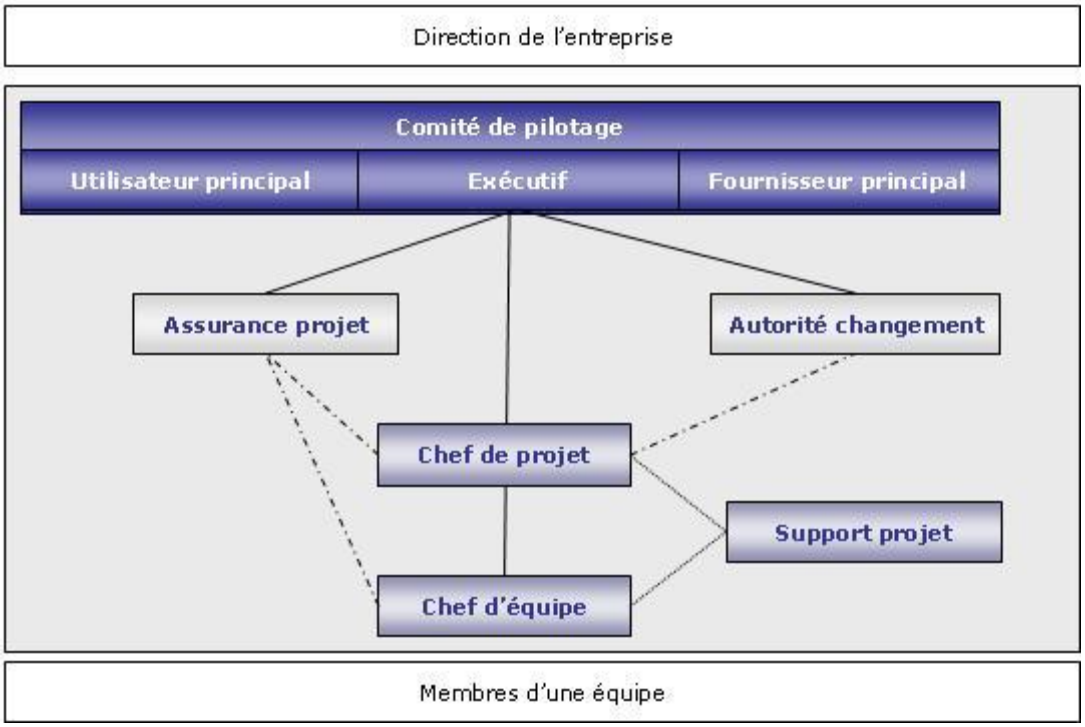
Gérer la livraison du produit (MP - Managing Product Delivery) dont l'objectif est de gérer la relation entre les équipes projet et le chef de projet, et notamment de garantir la bonne compréhension de part et d'autres du travail à effectuer :

- garantir que tous les travaux confiés à l'équipe ont bien été autorisés et assurer une compréhension commune des tâches à effectuer par tous les membres de l'équipe
- S'assurer du travail effectué, et de sa conformité aux exigences en matière de qualité
- Etablir des rapports sur l'avancement du projet et sur la qualité à l'attention du chef de projet
- Faire approuver les produits livrés

Clore le projet (CP - Closing a project) dont l'objectif est de clore proprement le projet et d'établir le bilan de fin de projet.

Focus sur l'organisation

Prince2 propose une organisation de pilotage projet type avec différents rôle :



- **Comité de pilotage**, qui assure le pilotage global du projet, et en réfère à la direction de l'entreprise. Ce comité de pilotage est composé de représentants des utilisateurs (la maîtrise d'ouvrage), de représentants des fournisseurs (maitrise d'oeuvre) et de « l'executive » qui pilote l'ensemble (la direction de projet).
- **Chef de projet** qui pilote opérationnellement chaque séquence du projet, et qui rapporte au comité de pilotage.
- **Chef d'équipe** qui pilote une équipe en charge d'un périmètre donné, est responsable de la bonne livraison des travaux, et rapporte au chef de projet.
- En complément à cette hiérarchisation des responsabilités projets, trois autres rôles complètent le dispositif projet
- **Assurance projet**, en charge de surveiller tous les aspects de performance et qualité du projet, indépendamment du chef de projet, et d'apporter du support au chef de projet si nécessaire. Cette assurance projet agit en délégation du comité de pilotage.
- **Autorité de changement**, qui a en charge d'approuver ou refuser les demandes de changements escaladées au comité de pilotage. Cette autorité de changement agit en délégation du comité de pilotage.
- **Support projet**, en charge d'assister les chefs d'équipe et de projets, dans des actions de management de projet (planification, tenue de tableau de risques par exemple...). Certaines organisations peuvent avoir un bureau de projet (PSO - Project Support Office) en charge d'assurer ce support aux chefs de projets.

Chapitre 10 : Scrum

10.1 Présentation

Scrum a été initié en 1995 par Ken Schwaber et Jeff Sutherland qui feront partie en 2001 des 17 signataires du Manifeste agile.

Les bases de Scrum sont définies dans un document de référence : le guide Scrum (http://www.scrumguides.org).

Scrum (n) : Un cadre de travail (framework) au sein duquel les acteurs peuvent aborder des problèmes complexes et adaptatifs, en livrant de manière efficace et créative des produits de la plus grande valeur possible.

Scrum est :

- Léger
- Simple à comprendre
- Difficile à maîtriser

Usage de Scrum :

Scrum a été initialement développé pour la gestion et le développement de produits. Depuis le début des années 1990, Scrum a été largement utilisé dans le monde entier pour :

- Rechercher et identifier des marchées, des technologies et des caractéristiques produit viables ;
- Développer des produits et des améliorations ;
- Publier des produits et des améliorations, jusqu'à plusieurs fois par jour ;
- Développer et maintenir des environnements Cloud (en ligne, sécurisé, à la demande) et d'autres environnements d'exploitation de produits ; et,
- Maintenir et renouveler des produits.

Scrum a été utilisé pour développer des logiciels, du matériel, des logiciels embarqués, des réseaux de fonctions interactives, des véhicules autonomes, des écoles, des gouvernements, du marketing de la gestion opérationnelle des organisations et presque tout ce que nous utilisons dans notre vie quotidienne.

10.2 Les fondamentaux

Objectifs de Scrum

Scrum met à disposition des moyens permettant de gérer des projets informatiques complexes de façon itérative. Chaque itération doit permettre :

- D'ajouter de la valeur au produit ;
- De préciser la définition du produit ;
- D'améliorer le processus de production.

Empirisme :

Scrum propose un contrôle empirique du processus de production logiciel. Il implique de baser sur l'expérience acquise pour améliorer l'organisation et le produit réalisé.

Il repose sur 3 piliers :

- **Transparence** par une communication constante et une compréhension partagée par tous les membres de l'équipe ;
- **Inspection** pour détecter les écarts par rapports aux attentes ;
- **Adaptation** pour réduire le risque et mettre en place une démarche d'amélioration.

Explications des trois piliers :

La **transparence** : les aspects importants du processus doivent être visibles à tous ceux qui sont responsables des résultats. La transparence requiert la définition d'un standard commun pour ces aspects afin que les observateurs partagent une compréhension commune de ce qui est observé.

L'inspection : les utilisateurs de Scrum doivent fréquemment inspecter les artefacts Scrum et l'état d'avancement par rapport à un Objectif de Sprint (Sprint Goal) afin de détecter les écarts indésirables. La fréquence de ces inspections ne devrait pas gêner le travail en cours. Ces inspections sont plus bénéfiques lorsqu'elles sont effectuées avec diligence par des inspecteurs qualifiés sur les lieux de travails.

Adaptation : si un inspecteur détermine qu'un ou plusieurs aspects du processus dérivent hors des limites acceptables, et que le produit qui en résulte ne sera pas acceptable, le processus ou le matériel utilisé par le processus doit être ajusté.

Scrum prescrit quatre événements formels d'inspection et d'adaptation, tels que décrit dans la section événements Scrum de ce document :

- Planification du Sprint (Sprint Planning)
- Mêlée Quotidienne (Daily Scrum)
- Revue de sprint (Sprint Review)
- Rétrospective de Sprint (Sprint Retrospective)

10.3 Rôles et acteurs

10.3.1 Scrum Master

Rôles :

- Responsable méthodologie
- Organisateur chargé d'accompagner les équipes dans l'utilisation des composantes de Scrum (événements, artefacts, etc).
- Facilitateur, aidant à fluidifier la communication

Compétences et qualités :

- Connaissance et expérience de Scrum
- Capacité à transmettre
- Intelligence relationnelle pour créer une synergie dans l'équipe

Au sein d'une équipe :

- Un seul Scrum Master par équipe Scrum
- Le rôle de Scrum Master peut être « tournant » : à chaque nouveau projet, un nouveau Scrum Master est désigné.

Les Scrum Masters remplissent leur rôle en aidant tout le monde à comprendre la théorie, les pratiques, les règles et les valeurs de Scrum.

Le Scrum Master est un leader-serviteur de l'équipe Scrum. Le Scrum Master assiste les personnes externes à l'équipe Scrum pour identifier quelles sont les interactions bénéfiques avec elle.

10.3.2 Le Product Owner (PO)

Rôles :

- Responsable du produit
- Validateur de toutes les décisions autour du produit et du contenu du carnet de produit
- Traducteur, il s'assure de la transparence (compréhension par tous) des besoins exprimés.

Compétences et qualités :

- Rigueur sur le maintien à jour des définitions du produit
- Capacité d'immersion pour assurer une compréhension optimale du métier et du besoin
- Ecoute et ouverture auprès des autres membres de l'équipe et de leurs propositions
- Esprit de synthèse

Au sein d'une équipe :

- Un seul PO par équipe Scrum
- Plaque tournante de toutes les discussions et décisions autour du produit

10.3.3 Equipe de développement

Rôles :

- Responsable de l'incrément, du carnet de sprint et donc de la manière dont sera réalisée l'application
- Chargé de l'estimation de la charge et du temps de réalisation des items
- Réalisation de la solution technique

Compétences et qualités :

L'équipe de développement est pluridisciplinaire. Elle regroupe toutes les compétences techniques nécessaires à la réalisation du logiciel : développeurs, architectes, concepteurs, etc.

Au sein d'une équipe :

Scrum préconise entre 3 et 9 membres pour optimiser la réactivité, la capacité à produire et la communication.

10.3.4 Les autres parties prenantes

Le Product Owner, le Scrum Master et l'équipe de développement constituent l'équipe **Core(noyau)** de Scrum. Celle-ci intervient sur le projet conjointement aux autres parties prenantes :

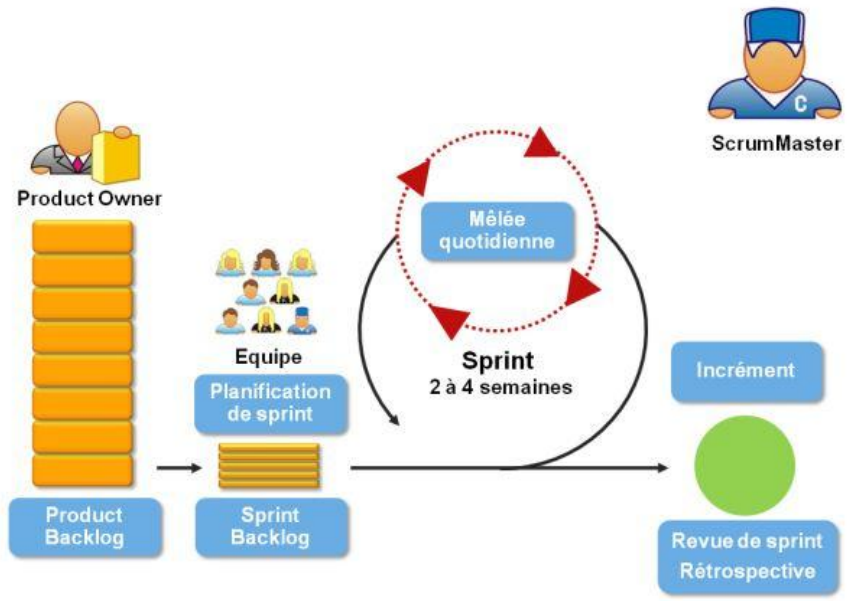
- Les **commanditaires** (clients) qui constituent l'initiateur du produit ;
- Les **utilisateurs** à qui est destiné l'outil ;
- Les **financeurs** (sponsor) qui apportent le support financier au projet.

10.4 Evénements

Les événements de Scrum entrent dans une logique de timebox.

Chaque de ces boîtes de temps doit respecter 2 principes :

- Limiter le temps à passer sur un événement donné ;
- N'accepter aucun dépassement.



10.5 Vision du produit et product backlog

La première étape consiste à formaliser la **vision du produit** (logiciel) que l'on souhaite réaliser. Cette vision décrit les principaux objectifs, jalons, utilisateurs visés. Elle contribuera à guider et fédérer les acteurs du projet. La suite consiste à établir la liste des **exigences fonctionnelles** et **non**

fonctionnelles du produit. Chaque exigence est ensuite estimée par l'**équipe de développement** avec la technique de **Planning Poker**. A la lueur des estimations, la liste ainsi complétée est ordonnancée. Les exigences seront converties en fonctionnalités utilisables selon cet ordonnancement. Le principe étant de convertir en premier les exigences qui apportent le plus de valeur ajoutée (ou ROI) au commanditaire. Il s'agit donc de faire remonter les exigences fonctionnelles de la plus haute valeur ajoutée (ou dont le ROI est le plus élevé) en haut de la liste. Cette liste est appelée le **Product Backlog**. Le Product Backlog servira à piloter l'équipe de développement et pourra évoluer tout au long du projet. **Le changement est non seulement autorisé mais encouragé afin de pouvoir éliminer les idées de départ qui s'avéreront mauvaises et de prendre en compte les nouvelles idées qui arriveront en cours de route.** Cette activité de construction du Product Backlog est collaborative, elle implique le Product Owner et l'équipe de développement.

Pondérer chaque exigence d'une valeur ajoutée n'est pas toujours évident pour une **équipe métier** (MOA) novice. Différentes techniques s'offrent à vous. Le fameux **Planning Poker** (voir paragraphe ci dessous) peut s'avérer utile pour déterminer collectivement les pondérations en « points » de valeur ajoutée. On peut également utiliser des échelles de valeur plus ou moins fine (exemple : « faible », « moyenne », « haute » ou encore les tailles de tee-shirt : XS, S, M, L, XL, XXL).

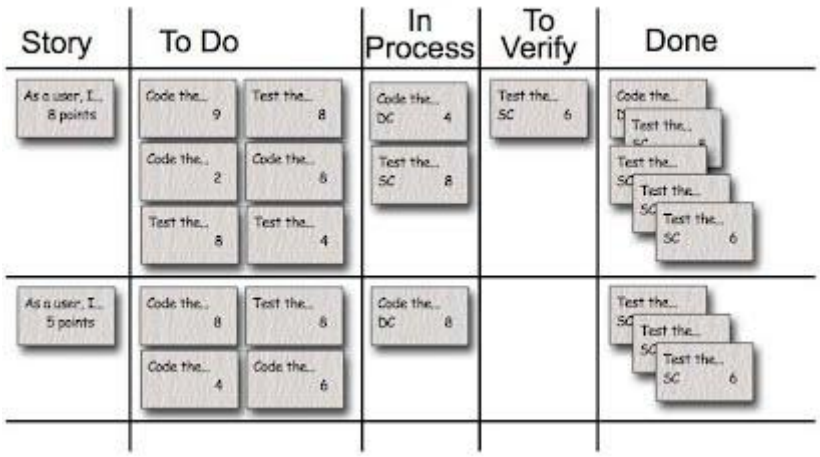
10.6 Réunion de planification de sprint

Durée maximum : 2 heures par semaine de sprint (autrement dit : 4 heures pour des sprints de 2 semaines).

Phase 1 : Le « Quoi »

Une fois que le **Product Backlog** est suffisamment complet et ordonnancé, on peut planifier un sprint. Le **Product Owner** revoit alors avec l'**équipe de développement** la **vision du produit**, la roadmap, le plan de livraison (jalons et deadline), l'objectif du sprint et le Product Backlog. L'équipe de développement vérifie les estimations, confirme qu'elles sont exactes et sélectionne en haut du Product Backlog les exigences qu'elle se sent capable de convertir en fonctionnalités utilisables d'ici la fin du sprint (il s'agit d'une prévision et non pas d'un engagement « contractuel »).

Phase 2 : Le « Comment »



Exemple de tableau des tâches

L'**équipe de développement** fait ensuite l'inventaire des tâches qui permettront de convertir les exigences sélectionnées en fonctionnalités utilisables d'ici la fin du sprint. Toutes les exigences n'ont pas nécessairement besoin d'être découpées en tâches. En cas de manque de temps, l'équipe de développement peut se contenter de découper celles qui seront réalisées au cours des premiers jours du sprint (elle découpera en cours de sprint les autres exigences). Elle doit cependant aller suffisamment loin dans l'effort de conception pour pouvoir vérifier sa prévision. Si elle constate après analyse des exigences sélectionnées, que sa prévision est erronée, elle peut réajuster avec le Product Owner la liste des exigences sélectionnées.

Les **tâches de développement** sont centralisées dans le **Sprint Backlog** et ajoutées au **tableau des tâches** physique (aussi appelé Kanban, même si Kanban veut dire bien plus). L'idéal est de parvenir à un découpage relativement fin (inférieur ou égal à une journée de travail).

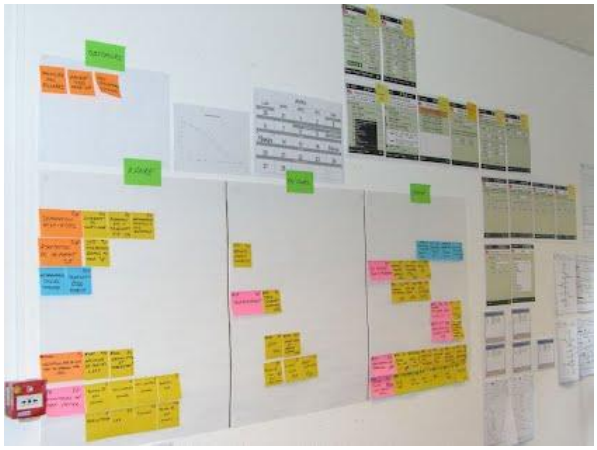


Photo d'un tableau des tâches

Chacun peut personnaliser les colonnes de son **tableau des tâches**, ou code couleur des post-it. Les **User Stories** (une **User Story** représente une exigence, voir annexe pour plus de détails) sont les gros post-it larges (rose et orange), les sous tâches des User Stories sont en jaune, on trouve en bleu des tâches imprévues indépendantes. Au dessus du tableau figurent les obstacles courants, un graphique d'avancement appelé **burdown chart** ainsi qu'un calendrier géant avec les dates clefs et les congés de chacun. Sur la droite on trouve des maquettes d'IHM, des documents métier, etc. Cette réunion de planification est l'occasion de préciser ou rappeler à l'équipe la **définition de « terminé »** pour une **user story** ou exigence. Exemple de définition de « terminé » : code commité, testé unitairement, documenté, testé en intégration, revue par un pair, **tests d'acceptation** de la user story passants.

Sprint (2 à 4 semaines)

Au cours du Sprint, l'équipe se concentre sur l'accomplissement des tâches du **Sprint Backlog**. En cas de retard (indiqué par le **Burndown Chart**), des exigences ou tâches seront retirées du Sprint Backlog en cours de route en essayant de préserver l'objectif du sprint (pour cela, il est conseillé d'ordonnancer les exigences au sein du sprint). Et inversement, si l'équipe avance plus vite que prévu, des exigences ou tâches y seront ajoutées. En accord avec le **Product Owner** dans les deux cas.

Les développements se font verticalement et non pas horizontalement par couche. Le but est de développer les fonctionnalités de bout en bout (de la conception aux tests) au fil de l'eau au cours du sprint. Autrement dit d'éviter un mini **cycle en V** au sein du sprint, voire de se retrouver avec une surcharge d'effort de test en fin de sprint. Les développeurs doivent donc éviter de trop paralléliser les exigences et encore moins les tâches de développement. Pour cela, le **pair programming** peut se révéler utile ainsi que la définition d'une limite maximum d'éléments au sein d'une colonne du tableau des tâches (voir notion de **Work In Progress** du Kanban). Si l'équipe de développement n'est pas convaincue, le « jeu du prénom en mode multitâche » peut s'avérer utile.

Le **tableau des tâches** physique rempli de post-it est pratiquement indispensable. Il permet d'avoir une vision claire du travail à accomplir, en cours et terminé. Il peut également s'avérer précieux lors des réunions quotidiennes, surtout si vous calculez à la main le « reste à faire » du Sprint afin de tracer le graphique d'avancement (voir plus loin le « Burndown Chart »). Le tableau facilite également l'affectation des tâches par l'équipe en ayant une vision d'ensemble du sprint en un coup d'œil. Inutile de se torturer à planifier à l'avance dans le détail l'activité de chaque développeur sur toute la durée du sprint. Il faut se détacher d'une approche prédictive et des diagrammes de Gantt et renoncer au **style de management « commander et contrôler »**. Ce sont les développeurs qui « tirent » les tâches et non pas le **Scrum Master** qui les affecte.

Pendant le sprint, l'équipe de développement assistera le **Product Owner** dans ses activités d'affinage du Product Backlog. Cette assistance peut consister en des ateliers de conception anticipés, de priorisation ou d'estimation. Il faut compter environ 10% de la capacité à faire de l'équipe de développement pour ces activités.

10.7 Mêlée quotidienne ou « stand-up meeting »

Durée maximum : 15 minutes.

Cette réunion qui se fait debout (ça évite de s'éterniser) est très importante. Elle permet quotidiennement aux membres de l'équipe de se synchroniser, de remonter les obstacles rencontrés, de s'entraider, de vérifier l'avancement du sprint. Elle contribue également à faire naître l'**esprit d'équipe**. A condition bien entendu de ne pas transformer cette réunion de « synchronisation » en réunion de « reporting » vers le Scrum Master.

Chaque personne répond à 3 questions :

- Qu'ai-je fait hier qui a aidé l'équipe de développement à atteindre l'objectif Sprint ?
- Que vais-je faire aujourd'hui pour aider l'équipe de développement à atteindre l'objectif Sprint ?
- Est ce que je vois des obstacles susceptibles de m'empêcher ou d'empêcher l'équipe de développement d'atteindre l'objectif du Sprint ?

Le **Scrum Master** est ainsi immédiatement au courant des obstacles rencontrés, il doit impérativement les prioriser, les suivre et bien sûr s'efforcer de les lever au plus tôt afin de garder l'équipe pleinement concentrée et productive.

La **mêlée quotidienne** se déroule à lieu et heure fixes (devant le tableau des tâches physique de préférence) déterminés par l'équipe de développement. Au début le Scrum Master peut avoir à rappeler qu'il est l'heure de la mêlée et animer cette dernière en rappelant les 3 questions et évitant l'instruction des problèmes ou obstacles en séance afin de ne pas dépasser les 15 minutes imparties. L'objectif du Scrum Master consiste cependant à viser l'appropriation de la mêlée par l'équipe de développement.

10.8 Graphique d'avancement (Burndown Chart)

Exemple de Sprint Backlog :

Sprint Backlog

Exigence	Sous Tâche	Reste A Faire									
Exigence X - #102	IHM	8	8	6							
	Mise à jour documentation	2	2	0							
	Services métier	10	4	0							
	Tests automatisés Perf	8	8	8							
Exigence YY - #103	IHM	8	8	8							
	Mise à jour documentation	2	2	2							
	Services métier	10	10	5							
	Tests automatisés Perf	8	8	8							
Exigence XY - #33	IHM	8	8	8							
	Mise à jour documentation	2	2	2							
	Services métier	10	10	7							
	Tests automatisés Perf	8	8	8							
Exigence Y - #365	IHM	8	8	4							
	Services métiers	10	5	0							
	Jours d'itération	Je	Ve	Lu	Ma	Me	Je	Ve	Lu	Ma	Me
	Trajectoire idéale	102	91	79	68	57	45	34	23	11	0
	Trajectoire réelle	102	91	66							

Pour connaître votre avancement, vous allez avoir besoin de tracer le **Burndown Chart** du sprint en cours. Ce graphique est simple, il s'agit du tracé de la charge de travail restante (généralement en heures) en fonction du temps (en jours). Pour tracer ce graphique, il suffit de mettre à jour (lors de chaque mêlée quotidienne par exemple) le **sprint backlog** (voir illustration).

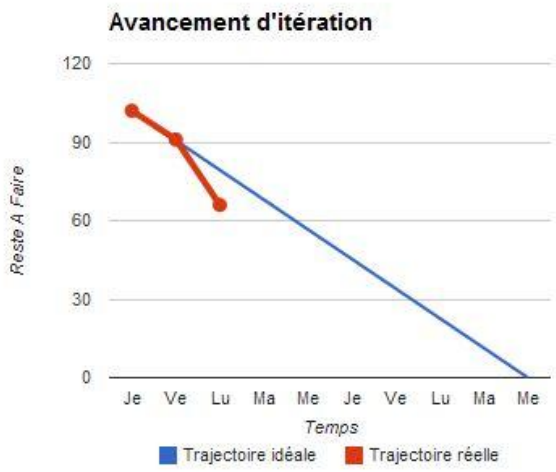


Figure 10: Exemple de Burndown Chart de Sprint

10.9 Revue de Sprint

Durée maximum : 1 heure par semaine de sprint (autrement dit : 2 heures pour des sprints de 2 semaines).

Fréquence : A la fin de chaque sprint.

L'objectif de la revue de sprint est d'inspecter l'incrément produit au cours du sprint écoulé, faire un point sur l'avancement de la Release et adapter au besoin le plan et le Product Backlog. L'équipe de développement présente à tout acteur projet intéressé (à minima le Product Owner idéalement accompagné d'utilisateurs finaux) les nouvelles fonctionnalités développées au cours du sprint. Le Product Owner donne un feedback à l'équipe de développement, il accepte ou refuse les fonctionnalités présentées.

L'équipe de développement calcule sa **vélocité** en additionnant les points d'estimations associées aux fonctionnalités acceptées. Une fonctionnalité partiellement terminée ne rapportera aucun point car une telle fonctionnalité n'est pas utilisable. La vélocité ainsi calculée va permettre de mettre à jour le graphique d'avancement de Release et de vérifier l'avancement de cette dernière. C'est l'occasion de vérifier que le nombre de sprints de la Release demeure adapté ou non.

La livraison à la fin de chaque sprint n'est pas obligatoire.

10.10 Rétrospective de sprint

Durée maximum : 45 minutes par semaine de sprint (autrement dit : 1 heure 30 pour des sprints de 2 semaines).

Fréquence : A la fin de chaque sprint.

Cette réunion est généralement animée par le « ScrumMaster » qui s'adresse à son équipe. Elle a pour but d'améliorer continuellement le processus de développement de l'équipe en mettant les idées de chacun à contribution. Il existe différentes techniques d'animation de rétrospectives. Voir le livre « Agile Retrospective: Making Good Teams Great » pour en savoir plus.

L'une d'elle consiste à identifier et pondérer les éléments positifs (éléments à cultiver ou source de motivation) du sprint écoulé, puis les éléments à améliorer, puis de définir un plan d'action d'amélioration (en commençant par améliorer les éléments dont la pondération est la plus forte). Si les membres de l'équipe sont à l'aise pour s'exprimer, un simple tour de table permet de remplir le paper-board d'éléments. Dans le cas contraire, on peut avoir recours aux post-it (chacun se voit remettre des post-it vierges et un marqueur et inscrit ses idées dessus pour ensuite les transmettre à l'animateur. Une idée par post-it.). Pour pondérer les éléments, il suffit d'allouer un certain nombre de points (5 par exemple) à chaque membre. Chaque membre peut ventiler ses points à sa guise (exemple : 4 points sur un sujet qu'il considère très important, 1 point sur un autre sujet également important à ses yeux mais quatre fois moins que le premier). Pendant la phase d'inventaire des éléments à améliorer, veillez à ne pas essayer de trouver des solutions avant la phase dédiée au plan d'action d'amélioration. Vous risqueriez de ne pas faire remonter à la surface la majorité des « problèmes ». Ce serait dommage car, j'ai pu constater que parfois, le simple fait d'évoquer un

problème sans forcément avoir le temps de trouver une solution entraîne une amélioration au sprint suivant.

Commencer par les points positifs peut être vital lorsque le sprint a été éprouvant. L'équipe aura peut être besoin de se nourrir des éléments positifs avant de s'attaquer aux éléments à améliorer.

Tous les domaines peuvent être abordés en rétrospective : humain, organisationnel, pratiques, processus, outillage, qualité de vie au travail, conflits, interactions avec le métier. Le compte rendu de la dernière rétrospective, les graphiques d'avancement de sprint et release, les événements du sprint, les feedbacks de la revue de sprint sont autant d'éléments qui alimentent les conversations.

Chapitre 11 : LSD- Lean Software Development

11.1 Pensée Lean

La base de la méthode Toyota est de ne pas satisfaire du statu quo, vous devez constamment vous poser la question « Pourquoi faisons-nous ça ? ».

Une philosophie :

- Respect des employés
- Une utilisation de toutes les compétences des employés
- Donner des responsabilités et avoir confiance dans les employés
- Amélioration continue
- Respect des personnes
- Remettre tout en cause
- Embrasser le changement

Nous pouvons faire petit historique des méthodes utiliser dans le passé.

- **Lean** (naissance de 1950-1960) : une pensée focus sur le flux de valeur, respect des personnes, amélioration continue ce qui résulte au Lean Startup.
- **Extreme Programming** (naissance de 1996-1998) : Pratiques d'Ingénierie logicielle et d'émancipation sociale ce qui résulte au Software Craftmanship, Devops.
- **Scrum** (naissance de 1993-1994, renaissance 2004-2006) : Framework de création de valeur dans un environnement complexe.
- **Kanban Software** (renaissance 2005-2009) : Système de gestion de flux de production et de management visuel poussant à l'amélioration continue qui résulte au Scrumban.

11.2 Les principes du Lean

Les principes du Lean sont compris de 7 points :

1. Éliminer les sources de gaspillage
2. Favoriser l'apprentissage
3. Reporter la décision
4. Livrer vite
5. Responsabiliser l'équipe
6. Construire la qualité intrinsèque
7. Optimiser le système dans son ensemble

Je vais faire une petite explication des 7 points ci-dessus :

a. Eliminer les sources de gaspillages

1 Travail partiellement fait

Du stock qui ne rapporte rien, qui nous rend moins réactif et qu'on doit donc absolument réduire … Avec le travail partiellement fait, on ne sait jamais si ça marche ! C'est donc risqué

et ça peut devenir obsolète. Pure perte et danger, sans compter le temps pour revenir dessus !

Le développement itératif et incrémental, le « timeboxing[15]» , le travail découpé, morcelé et des outils comme le Kanban board et des règles simples améliorent les choses. Partager une vision commune du « Done » et mettre en avant est essentiel: un sprint se termine par une demo et une livraison, c'est la base.

2 Process inutiles

Chaque process, chaque action engagée doit être source de valeur pour le Client et doit être attendu par les équipes qui en sont les destinataires. Lean nous propose un outil remarquable : la Value Stream Map, outil idéal pour chasser les process inutiles et point d'entrée inégalé pour évaluer les process de développement…
Regardez autour de vous… faites un point rapide sur votre activité, sur un projet; tracez par exemple demande Client, depuis son arrivée dans l'organisation jusqu'à son traitement effectif (client livré …), repèrez les temps d'attente qui ne rapportent rien et les activités sources de valeur, c'est trés instrcutif, parfois désolant …
Agile Tip !
Des méthodes & process simples comme SCRUM vous permettent de limiter les risques; de fréquentes rétrospectives vous permettront d'ajuster. La Value Stream Map[16] vous guidera.

3 Excès de fonctionnalités

C'est le pire !

Le développement de fonctions non attendues, non utilisées est le plus gros gaspillage. Le standish Group a pourtant multiplié les rapports depuis une quinzaine d'années: rien n'y fait! ***Simplicité et « Juste ce qu'il faut »***, c'est un message clé de mon blog. ***Agile Tip !***

[15] La **gestion par blocs de temps,** ou **méthode du temps limité**[2] (en anglais « ***timeboxing* »**) est une approche de planification et de gestion de temps qui consiste à allouer à la réalisation d'une activité donnée une durée fixe, volontairement limitée de temps.

[16] ***Value stream mapping*** (VSM), ou ***Cartographie des chaines de valeur*** en français, est une méthode de lean manufacturing de cartographie d'un processus. Le *value stream mapping* ou VSM est un outil regroupant toutes les actions (à valeur ajoutée et à non valeur ajoutée) qui amènent un produit d'un état initial à un état final. Cet outil s'attache à travailler sur un ensemble et non une partie : sur une ligne de production, le VSM ne s'attaque pas à une machine de la ligne en particulier mais à l'ensemble de celles-ci.

Le but de cette cartographie est d'obtenir une vision simple et claire d'un processus. La plupart du temps le VSM se limite à l'entreprise même, mais il peut être bon d'incorporer l'approvisionnement en amont et la livraison aux clients en aval.

Le but ultime du VSM est d'avoir une représentation du temps de défilement (lead time) du produit choisi. L'analyse amènera ensuite des améliorations qui porteront sur la globalité du processus et non sur une étape lambda.

L'analyse de Kano, l'intégration d'une démarche ergonomique et les feedbacks utilisateur et encore l'effort de priorisation demandé au Métier peuvent vous aider à y voir plus clair, et à concrétiser un backlog de produit, digne de ce nom

4 Passages d'une tâche à une autre

On le sait tous, cumuler les tâches et les projets est coûteux d'un point de vue cognitif, outre la surchage potentielle, c'est le temps qu'il faut pour se remettre en contexte qui constitue du temps gaspillé pour notre Client.

Agile Tip !

Une équipe intégrée, à plein temps, des sprints (itérations) courts, des règles simples comme limiter le « In progress » réduisent ce risque de gaspillage

5 Attentes et retards

Le temps pour démarrer, staffer, valider, décider, les retards dans les tests, dans le déploiement, dans l'analyse… pour au final retarder la livraison finale.

Ce qui nous pénalise : des cycles de temps qui ne dépendent pas de nous mais qui nous mettent dans une position d'attente, ou d'autres cycles dont nous sommes responsables. L'ensemble va retarder l'atteinte de nos seuls objectifs: donner exactement à notre Client ce qu'il attend de nous, où il le veut et quand il le veut, et lui apporter de la valeur au travers de nos activités.

Le gaspillage de temps n'est pas la seule conséquence… insatisfaction, livraison obsolète (le client a changé d'avis sur telle ou telle partie), livraison nulle (le client a changé d'avis tout court, et ne nous a pas attendu), produit obsolète (le marché ne nous a pas attendu !!)

Agile Tip !

Une approche itérative et incrémentale avec des cycles courts permet un feedback et une adaptation permanente et facilite les réponses au changement. Un backlog de produit estimé, priorisé et maintenu donne de la visibilité et permet de réagir vite. Le Kanban board accentuera la fluidité, et plus largement le radiateur d'informations permettra de savoir où on en est et d'identifier le cas échéant les goulets d'étranglement.

6 Transmission d'informations

Le temps pour obtenir une réponse, l'accès au client, aux équipes… il peut s'agir en premier lieu des déplacements de personnes, mais cela concerne aussi la perdition d'information qui nous guette à chaque transmission de doc. C'est un vrai problème pour les équipes distribuées…

Petit test : entre une explication en face à face, live au tableau blanc d'une interaction et de la navigation entre quelques écrans clés, et les mêmes explications, à distance, au tél (pire par mail), quel est le plus efficace ?

Proximité, war Room, équipe intégrées, ateliers de travail, « juste ce qu'il faut » de documentation sont des bonnes pratiques agiles qui peuvent aider…

7 Défauts

Les défauts, anomalies sont aussi, dans l'esprit Lean, **des stocks à éliminer**. Une anomalie n'a pas de valeur. L'accumulation (files d'attentes ingérables et surchargées…) ou une détection trop tardive, feront exploser les couts et les délais, sans compter la satisfaction Client et l'image que vous renvoyez.

Aujourd'hui, la qualité n'est plus négociable et les attentes sont fortes. TDD, tests unitaires, intégration continue, standards et conventions, revues de code … bref toutes les pratiques d'ingénierie eXtreme Programming mais aussi le Feedback permanent du client, le développement itératif, un focus sur la qualité, des techniques comme les 5 Whys améliorent sensiblement la situation

b. Favoriser l'apprentissage

On effectue des itérations et des Feedback.

c. Reporter la décision

Reporter la décision, c'est le **3ème** principe du Lean Software Developement, un principe qui va à l'encontre de pas mal d'idées reçues quant à la façon de prendre des décisions, de gérer les projets, et même d'envisager l'apprentissage…

Le Lean Software Development nous invite en effet à laisser au maximum **les options ouvertes** et à **minimiser les actions potentiellement irréversibles**.
C'est finalement reporter la décision jusqu'au dernier moment raisonnable (mais attention la nuance, **pas le dernier moment**), et concrètement éviter de discuter, de perdre du temps sur des éléments qu'on ne veut pas traiter tout de suite.

d. Livrer vite

Une fois de plus, Lean et Méthodes Agiles convergent fortement… d'ailleurs une **mise en marché** plus rapide est l'une des principales raisons pour lesquelles les organisations choisissent de basculer vers l'Agilité.

Livrer vite, principe Lean, principe Agile que Google porte haut et fort…

e. Responsabiliser l'Equipe

Responsabiliser l'Equipe, c'est le 5ème principe du Lean Software Development; c'est aussi un principe de management que beaucoup d'entreprises se vantent d'appliquer, **en théorie**… **sur les plaquettes**… mais qui concrètement est rarement suivi des faits et se retrouve peu sur le terrain ! Car **responsabiliser l'Equipe** n'est pas sans conséquences, l'affaire est complexe, et nécessite entre autres de revoir les modèles organisationnels existants, d'envisager différemment la conduite de projet, d'équilibrer les forces et de perdre un peu le contrôle…

Ce que le **Lean** nous dit :

- Un nouveau postulat : **les personnes savent travailler**, connaissent leur boulot, nous devons chercher avant tout à changer l'environnement qui les empêche de travailler. Une position très ergonomique.

  Recherchons l'amélioration continue par la participation, les RDV collectifs, l'écoute, par de fréquentes rétrospectives ; et soyons réactifs mais précis quant aux réponses que nous apportons. Etablissons des plans d'action simples mais efficaces et partagés **Transparence et confiance**.

- Donnons des **objectifs** (SMART…) à nos équipes et jouons enfin sur les vrais leviers motivationnels. Allons plus loin sur le sentiment d'appartenance, sur la compétence et le progrès !

- Cherchons **les leaders**, favorisons **le leadership** (c'est à dire *la capacité d'une personne à influencer, motiver et à rendre les autres capables de contribuer à l'efficacité et au succès de l'organisation dont ils sont membres*). Les entreprises sont remplies de gestionnaires, de managers, à tous les étages, à tous les niveaux … **combien de vrais leaders ?**

- Cultivons **l'expertise** (notamment celle que la concurrence n'a pas), par exemple en favorisant l'apprentissage.

f. Construire la qualité

g. Optimiser le système dans son ensemble

C'est 7ème et dernier principe fondateur du Lean Software development qui tient en une seule expression « **System thinking**« , une attitude **holistique** finalement intimement liée aux six principes précédents !

Le Lean Software Development nous incite ici à nous positionner sur une vue **Produit & Valeur** plutôt que strictement projet, à penser **système et objectifs**, notamment sur les aspects contrats et surtout **mesures**, deux outils cruciaux, au cœur de l'approche Lean.

Le Lean Software Development conseille de se focaliser sur **3 mesures essentielles**, dans lesquelles les différents acteurs du projet se retrouveront:

- Temps de cycle, appréhendé par la Value Stream Map
- ROI
- Satisfaction Client

Partie 3 : Modélisations

Chapitre 1 : UML

1.1 Signification

UML signifie Unified Modified Language en français « langage de modélisation objet unifié ». L'UML est type de modèle utiliser dans le concept de la programmation orientée Objet (POO).

UML est à la fois une norme, un langage de modélisation objet, un support de communication et un cadre méthodologique.

1.2 UML est une norme

Fin 1997, UML est devenu une norme OMG (Object Management Group).
L'OMG est un organisme à but non lucratif, créé en 1989 à l'initiative de grandes sociétés (HP, Sun, Unisys, American Airlines, Philips...). Aujourd'hui, l'OMG fédère plus de 850 acteurs du monde informatique. Son rôle est de promouvoir des standards qui garantissent l'interopérabilité entre applications orientées objet, développées sur des réseaux hétérogènes.
L'OMG propose notamment l'architecture CORBA (Common Object Request Broker Architecture), un modèle standard pour la construction d'applications à objets distribués (répartis sur un réseau).
CORBA fait partie d'une vision globale de la construction d'applications réparties, appelée OMA (Object Management Architecture) et définie par l'OMG. Sans rentrer dans les détails, on peut résumer cette vision par la volonté de favoriser l'essor industriel des technologies objet, en offrant un ensemble de solutions technologiques non propriétaires, qui suppriment les clivages techniques.
UML a été adopté (normalisé) par l'OMG et intégré à l'OMA, car il participe à cette vision et parce qu'il répond à la "philosophie" OMG.

1.3 UML est un langage de modélisation objet

Pour penser et concevoir objet, il faut savoir "prendre de la hauteur", jongler avec des concepts abstraits, indépendants des langages d'implémentation et des contraintes purement techniques.
Les langages de programmation ne sont pas un support d'analyse adéquat pour "concevoir objet". Ils ne permettent pas de décrire des solutions en termes de concepts abstraits et constituent un cadre trop rigide pour mener une analyse itérative.
Pour conduire une analyse objet cohérente, il ne faut pas directement penser en termes de pointeurs, d'attributs et de tableaux, mais en termes d'association, de propriétés et de cardinalités...
Utiliser le langage de programmation comme support de conception ne revient bien souvent qu'à juxtaposer de manière fonctionnelle un ensemble de mécanismes d'implémentation, pour résoudre un problème qui nécessite en réalité une modélisation objet.
L'approche objet nécessite une analyse réfléchie, qui passe par différentes phases exploratoires.
Bien que raisonner en termes d'objets semble naturel, l'approche fonctionnelle reste la plus intuitive pour nos esprits cartésiens... Voilà pourquoi il ne faut pas se contenter d'une implémentation objet, mais se discipliner à "penser objet" au cours d'une phase d'analyse préalable.
Toutes les dérives fonctionnelles de code objet ont pour origine le non respect des concepts de base de l'approche objet (encapsulation...) ou une utilisation détournée de ces concepts (héritage sans classification...). Ces dérives ne sont pas dues à de mauvaises techniques de programmation ; la racine du mal est bien plus profonde : programmer en C++ ou en Java n'implique pas forcément concevoir objet...
Les difficultés de mise en œuvre d'une approche "réellement objet" ont engendré bien souvent des déceptions, ce qui a longtemps constitué un obstacle important à l'essor des technologies objet.

Beaucoup ont cédé au leurre des langages de programmation orientés objet et oublié que le code n'est qu'un "moyen". Le respect des concepts fondamentaux de l'approche objet prime sur la manière dont on les implémente. Ne penser qu'à travers un langage de programmation objet détourne de l'essentiel.
Pour sortir les technologies objet de cette impasse, l'OMG propose UML.
UML comble une lacune importante des technologies objet. Il permet d'exprimer et d'élaborer des modèles objet, indépendamment de tout langage de programmation. Il a été pensé pour servir de support à une analyse basée sur les concepts objet. UML est un langage formel, défini par un métamodèle.
Le métamodèle d'UML décrit de manière très précise tous les éléments de modélisation (les concepts véhiculés et manipulés par le langage) et la sémantique de ces éléments (leur définition et le sens de leur utilisation).
En d'autres termes : UML normalise les concepts objet.
Un métamodèle permet de limiter les ambiguïtés et encourage la construction d'outils. Il permet aussi de classer les différents concepts du langage (selon leur niveau d'abstraction ou leur domaine d'application) et expose ainsi clairement sa structure. Enfin, on peut noter que le métamodèle d'UML est lui-même décrit par un méta-métamodèle de manière standardisée, à l'aide de MOF (Meta Object Facility : norme OMG de description des métamodèles).
Véritable clé de voûte de l'OMA, UML est donc un outil indispensable pour tous ceux qui ont compris que programmer objet, c'est d'abord concevoir objet. UML n'est pas à l'origine des concepts objets, mais il en constitue une étape majeure, car il unifie les différentes approches et en donne une définition plus formelle.

1.4 Le contexte d'apparition d'UML

1.4.1 Un approche fonctionnelle

LA DECOUPE FONCTIONNELLE D'UN PROBLEME INFORMATIQUE : UNE APPROCHE INTUITIVE

La découpe fonctionnelle d'un problème (sur laquelle reposent les langages de programmation structurée) consiste à découper le problème en blocs indépendants. En ce sens, elle présente un caractère intuitif fort.

LA REUTILISABILITE DU CODE

Le découpage d'un problème en blocs indépendants (fonctions et procédures) va permettre aux programmeurs de réutiliser les fonctions déjà développées (à condition qu'elles soient suffisamment génériques). La productivité se trouve donc accrue.

LE REVERS DE LA MEDAILLE : MAINTENANCE COMPLEXE EN CAS D'EVOLUTION

Le découpage en blocs fonctionnels n'a malheureusement pas que des avantages. Les fonctions sont devenues interdépendantes : une simple mise à jour du logiciel à un point donné, peut impacter en cascade une multitude d'autres fonctions. On peut minorer cet impact, pour peu qu'on utilise des fonctions plus génériques et des structures de données ouvertes. Mais respecter ces contraintes rend l'écriture du logiciel et sa maintenance plus complexe.
En cas d'évolution majeure du logiciel (passage de la gestion d'une bibliothèque à celle d'une médiathèque par exemple), le scénario est encore pire. Même si la structure générale du logiciel

reste valide, la multiplication des points de maintenance, engendrée par le chaînage des fonctions, rend l'adaptation très laborieuse. Le logiciel doit être retouché dans sa globalité :

- on a de nouvelles données à gérer (ex : DVD)
- les traitements évoluent : l'affichage sera différent selon le type (livre, CD, DVD …)

PROBLEMES GENERES PAR LA SEPARATION DES DONNEES ET DES TRAITEMENTS :

Examinons le problème de l'évolution de code fonctionnel plus en détail...
Faire évoluer une application de gestion de bibliothèque pour gérer une médiathèque, afin de prendre en compte de nouveaux types d'ouvrages (cassettes vidéo, CD-ROM, etc...), nécessite :

- de faire évoluer les structures de données qui sont manipulées par les fonctions,
- d'adapter les traitements, qui ne manipulaient à l'origine qu'un seul type de document (des livres)

Il faudra donc modifier toutes les portions de code qui utilisent la base documentaire, pour gérer les données et les actions propres aux différents types de documents.
Il faudra par exemple modifier la fonction qui réalise l'édition des "lettres de rappel" (une lettre de rappel est une mise en demeure, qu'on envoie automatiquement aux personnes qui tardent à rendre un ouvrage emprunté). Si l'on désire que le délai avant rappel varie selon le type de document emprunté, il faut prévoir une règle de calcul pour chaque type de document.
En fait, c'est la quasi-totalité de l'application qui devra être adaptée, pour gérer les nouvelles données et réaliser les traitements correspondants. Et cela, à chaque fois qu'on décidera de gérer un nouveau type de document !

1ERE AMELIORATION : RASSEMBLER LES VALEURS QUI CARACTERISENT UN TYPE, DANS LE TYPE

Une solution relativement élégante à la multiplication des branches conditionnelles et des redondances dans le code (conséquence logique d'une trop grande ouverture des données), consiste tout simplement à centraliser dans les structures de données, les valeurs qui leurs sont propres. Par exemple, le délai avant rappel peut être défini pour chaque type de document. Cela permet donc de créer une fonction plus générique qui s'applique à tous les types de documents.

2EME AMELIORATION : CENTRALISER LES TRAITEMENTS ASSOCIES A UN TYPE, AUPRES DU TYPE

Pourquoi ne pas aussi rassembler dans une même unité physique les types de données et tous les traitements associés ?
Que se passerait-il par exemple si l'on centralisait dans un même fichier, la structure de données qui décrit les documents et la fonction de calcul du délai avant rappel ? Cela nous permettrait de retrouver immédiatement la partie de code qui est chargée de calculer le délai avant rappel d'un document, puisqu'elle se trouve au plus près de la structure de données concernée.
Ainsi, si notre médiathèque devait gérer un nouveau type d'ouvrage, il suffirait de modifier une seule fonction (qu'on sait retrouver instantanément), pour assurer la prise en compte de ce nouveau type de document dans le calcul du délai avant rappel. Plus besoin de fouiller partout dans le code...
Ecrit en ces termes, le logiciel serait plus facile à maintenir et bien plus lisible. Le stockage et le calcul du délai avant rappel des documents, serait désormais assuré par une seule et unique unité physique (quelques lignes de code, rapidement identifiables).
Pour accéder à la caractéristique "délai avant rappel" d'un document, il suffit de récupérer la valeur correspondante parmi les champs qui décrivent le document. Pour assurer la prise en compte d'un nouveau type de document dans le calcul du délai avant rappel, il suffit de modifier une seule fonction, située au même endroit que la structure de données qui décrit les documents.

Document
Code document Nom document Type document
Calculer date rappel

Centraliser les données d'un type et les traitements associés, dans une même unité physique, permet de limiter les points de maintenance dans le code et facilite l'accès à l'information en cas d'évolution du logiciel.

1.4.2 L'approche objet

LE CONCEPT D'OBJET

Les modifications qui ont été apportées au logiciel de gestion de médiathèque, nous ont amené à transformer ce qui était à l'origine une structure de données, manipulée par des fonctions, en une entité autonome, qui regroupe un ensemble de propriétés cohérentes et de traitements associés. Une telle entité s'appelle... un objet et constitue le concept fondateur de l'approche du même nom.
Un objet est une entité aux frontières précises qui possède une identité (un nom).
Un ensemble d'attributs caractérise l'état de l'objet.
Un ensemble d'opérations (méthodes) en définissent le comportement.
Un objet est une instance de classe (une occurrence d'un type abstrait).
Une classe est un type de données abstrait, caractérisé par des propriétés (attributs et méthodes) communes à des objets et permettant de créer des objets possédant ces propriétés.

Document
+ code document : int + nom document : String + type document : String
+ Calculer date rappel () : Date

Classe : regroupement d'objets

MERISE_UML_document
C1 De Merise vers UML Support de cours

LES AUTRES CONCEPTS IMPORTANTS DE L'APPROCHE OBJET.

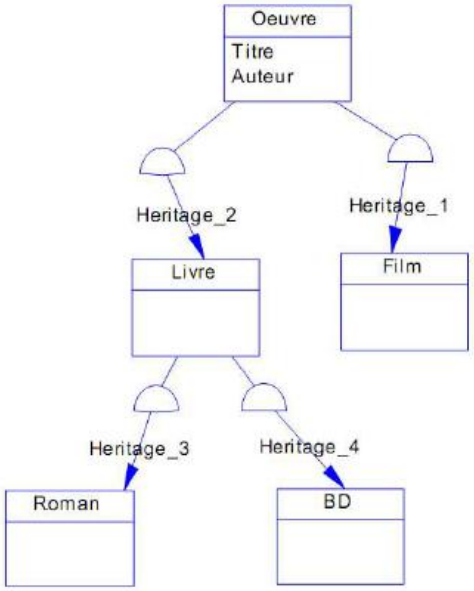
- **l'encapsulation**

L’encapsulation consiste à masquer les détails d'implémentation d'un objet, en définissant une interface.
L'interface est la vue externe d'un objet, elle définit les services accessibles (offerts) aux utilisateurs de l'objet.
L'encapsulation facilite l'évolution d'une application car elle stabilise l'utilisation des objets : on peut modifier l'implémentation des attributs d'un objet sans modifier son interface.
L'encapsulation garantit l'intégrité des données, car elle permet d'interdire l'accès direct aux attributs des objets.

- **l’héritage**

L'héritage est un mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe.
Une classe peut être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques ou d'en adapter certaines.

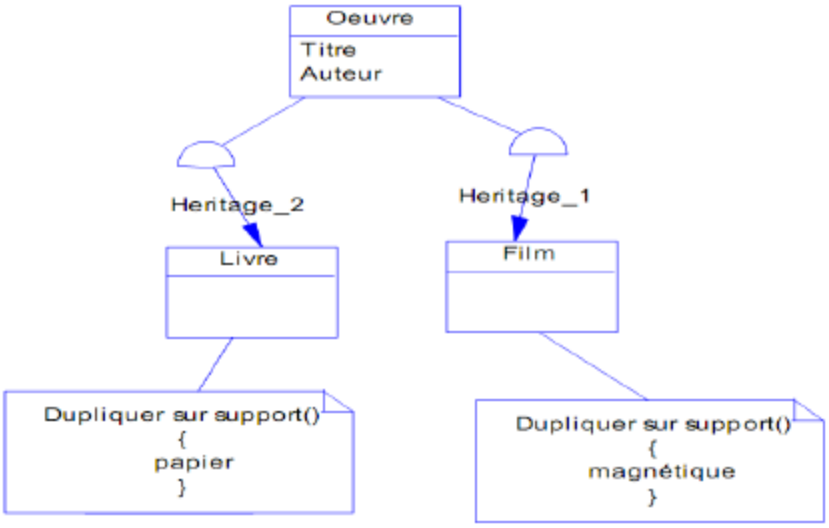
Plusieurs classes peuvent être généralisées en une classe qui les factorise, afin de regrouper les caractéristiques communes d'un ensemble de classes.
La spécialisation et la généralisation permettent de construire des hiérarchies de classes.
L'héritage peut être simple ou multiple.
L'héritage évite la duplication et encourage la réutilisation



- **Le polymorphisme**

Le polymorphisme représente la faculté d'une même opération de s'exécuter différemment suivant le contexte de la classe où elle se trouve.
Ainsi, une opération définie dans une superclasse peut s'exécuter de façon différente selon la sousclasse où elle est héritée.
Ex : exécution d'une opération de calcul des salaires dans 2 sous-classes spécialisées : une pour les cadres, l'autre pour les non-cadres.

Le polymorphisme augmente la généricité du code.

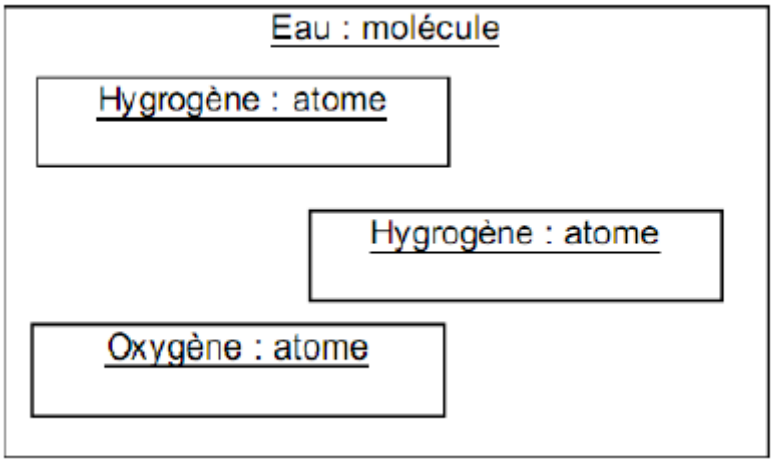


• **l'agrégation**

Il s'agit d'une relation entre deux classes, spécifiant que les objets d'une classe sont des composants de l'autre classe.

Une relation d'agrégation permet donc de définir des objets composés d'autres objets.

L'agrégation permet d'assembler des objets de base, afin de construire des objets plus complexes.



1.4.3 La genèse d'UML

LES PREMIERES METHODES D'ANALYSE (ANNEES 70)

Découpe cartésienne (fonctionnelle et hiérarchique) d'un système.

L'APPROCHE SYSTEMIQUE (ANNEES 80)

Modélisation des données + modélisation des traitements (Merise ...).

L'EMERGENCE DES METHODES OBJET (1990-1995)

Prise de conscience de l'importance d'une méthode spécifiquement objet :

- comment structurer un système sans centrer l'analyse uniquement sur les données ou uniquement sur les traitements (mais sur les deux) ?

- Plus de 50 méthodes objet sont apparues durant cette période (Booch, Classe-Relation, Fusion, HOOD, OMT, OOA, OOD, OOM, OOSE...)!
- Aucune méthode ne s'est réellement imposée.

LES PREMIERS CONSENSUS (1995)

- OMT (James Rumbaugh) : vues statiques, dynamiques et fonctionnelles d'un système
    - Issue du centre de R&D de General Electric.
    - Notation graphique riche et lisible.
- OOD (Grady Booch) : vues logiques et physiques du système
    - Définie pour le DOD, afin de rationaliser de développement d'applications ADA, puis C++.
    - Ne couvre pas la phase d'analyse dans ses 1ères versions (préconise SADT).
    - Introduit le concept de package (élément d'organisation des modèles).
- OOSE (Ivar Jacobson) : couvre tout le cycle de développement
    - Issue d'un centre de développement d'Ericsson, en Suède.
    - La méthodologie repose sur l'analyse des besoins des utilisateurs

L'UNIFICATION ET LA NORMALISATION DES METHODES (1995-1997)

En octobre 1994, G. Booch (père fondateur de la méthode Booch) et J. Rumbaugh (principal auteur de la méthode OMT) ont décidé de travailler ensemble pour unifier leurs méthodes au sein de la société Rational Software. Un an après, I. Jacobson (auteur de la méthode OOSE et des cas d'utilisation) a rejoint Rational Software pour travailler sur l'unification. Unified Modified Language (UML) est né.

Les travaux sur ce langage ont continué avec son adoption par de grands acteurs industriels comme HP, Microsoft, Oracle ou Unisys. Ce travail a abouti en 1997 à UML 1.0. Le langage a été soumis par Rational Software et ses partenaires à l'OMG comme réponse à un appel d'offres sur la standardisation des langages de modélisation.

L'appel d'offres de l'OMG a recueilli un avis favorable, puisque 6 réponses concurrentes sont parvenues à l'OMG. IBM et Object Time (méthode ROOM pour les systèmes temps réel réactifs) ont décidé de rejoindre l'équipe UML ; leur proposition était en fait une extension d'UML 1.0.

Certains autres auteurs qui ont répondu à l'appel d'offres ont abandonné leur proposition pour rejoindre à leur tour UML. En novembre 1997, UML a été adopté par l'OMG.

UML est donc le résultat d'un large consensus et tient compte des dernières avancées en matière de modélisation et de développement logiciel.

L'OMG RTF (nombreux acteurs industriels) centralise et normalise les évolutions d'UML au niveau international et de nombreux groupes d'utilisateurs UML favorisent le partage des expériences.

1.5 Les vues

LA VUE LOGIQUE

Cette vue concerne « l'intégrité de conception ».

Cette vue de haut niveau se concentre sur l'abstraction et l'encapsulation, elle modélise les éléments et mécanismes principaux du système.

Elle identifie les éléments du domaine, ainsi que les relations et interactions entre ces éléments « notions de classes et de relations » :

- les éléments du domaine sont liés au(x) métier(s) de l'entreprise,
- ils sont indispensables à la mission du système,
- ils gagnent à être réutilisés (ils représentent un savoir-faire).

Cette vue organise aussi (selon des critères purement logiques), les éléments du domaine en "catégories" :

- pour répartir les tâches dans les équipes,
- regrouper ce qui peut être générique,

- isoler ce qui est propre à une version donnée, etc...

LA VUE DES COMPOSANTS

Cette vue concerne « l'intégrité de gestion ».
Elle exprime la perspective physique de l'organisation du code en termes de modules, de composants et surtout des concepts du langage ou de l'environnement d'implémentation.
Dans cette perspective, l'architecte est surtout concerné par les aspects de gestion du code, d'ordre de compilation, de réutilisation, d'intégration et d'autres contraintes de développement pur.
Pour représenter cette perspective, UML fournit des concepts adaptés tels que les modules, les composants, les relations de dépendance, l'interface …
Cette vue de bas niveau (aussi appelée « vue de réalisation »), montre ainsi :

- l'allocation des éléments de modélisation dans des modules (fichiers sources, bibliothèques dynamiques, bases de données, exécutables, etc...). Cette vue identifie les modules qui réalisent (physiquement) les classes de la vue logique.
- l'organisation des composants, c'est-à-dire la distribution du code en gestion de configuration, les dépendances entre les composants...
- les contraintes de développement (bibliothèques externes...).
- l'organisation des modules en "sous-systèmes", les interfaces des sous-systèmes et leurs dépendances (avec d'autres sous-systèmes ou modules).

LA VUE DES PROCESSUS

Cette vue concerne « l'intégrité d'exécution ».
Cette vue est très importante dans les environnements multitâches ; elle exprime la perspective sur les activités concurrentes et parallèles. Elle montre ainsi :
- la décomposition du système en termes de processus (tâches).
- les interactions entre les processus (leur communication).
- la synchronisation et la communication des activités parallèles (threads).

LA VUE DE DEPLOIEMENT

Cette vue concerne « l'intégrité de performance ». Elle exprime la répartition du système à travers un réseau de calculateurs et de nœuds logiques de traitements. Cette vue est particulièrement utile pour décrire la distribution d'un système réparti.
Elle montre :
- la disposition et nature physique des matériels, ainsi que leurs performances.
- l'implantation des modules principaux sur les nœuds du réseau.
- les exigences en termes de performances (temps de réponse, tolérance aux fautes et pannes...).

LA VUE DES CAS D'UTILISATION

Cette vue est particulière en ce sens qu'elle guide toutes les autres.
Cette vue permet :
- de trouver le « bon » modèle

Les cas d'utilisation permettent de guider la modélisation. L'utilisation des scénarios et des cas d'utilisation s'avère plus rigoureuse et plus systématique que les entretiens et l'analyse des documents pour découvrir les abstractions du domaine.
- d'expliquer et de justifier ses choix

Il est en effet nécessaire d'expliquer le système, de justifier les choix qui ont guidé sa conception et son fonctionnement pour pouvoir le construire, le maintenir et le tester. Pour cela UML offre des concepts adaptés tels que les scénarios et les cas d'utilisation.

1.6 Les niveaux d'abstraction

- **Conceptualisation**

L'entrée de l'analyse à ce niveau est le dossier d'expression des besoins client. A ce niveau d'abstraction, on doit capturer les besoins principaux des utilisateurs.
Il ne faut pas chercher l'exhaustivité, mais clarifier, filtrer et organiser les besoins.
Le but de la conceptualisation est :

- de définir le contour du système à modéliser (de spécifier le "quoi"),
- de capturer les fonctionnalités principales du système, afin d'en fournir une meilleure compréhension (le modèle produit sert d'interface entre les acteurs du projet),
- de fournir une base à la planification du projet.

- **Analyse du domaine**

L'entrée de l'analyse à ce niveau, est le modèle des besoins clients (les "cas d'utilisation" UML).
Il s'agit de modéliser les éléments et mécanismes principaux du système.
On identifie les éléments du domaine, ainsi que les relations et interactions entre ces éléments :

- les éléments du domaine sont liés au(x) métier(s) de l'entreprise,
- ils sont indispensables à la mission du système,
- ils gagnent à être réutilisés (ils représentent un savoir-faire).

A ce stade, on organise aussi (selon des critères purement logiques), les éléments du domaine en "catégories", pour répartir les tâches dans les équipes, regrouper ce qui peut être générique, etc...

- **Analyse applicative**

A ce niveau, on modélise les aspects informatiques du système, sans pour autant rentrer dans les détails d'implémentation.
Les interfaces des éléments de modélisation sont définis (cf. encapsulation). Les relations entre les éléments des modèles sont définies.
Les éléments de modélisation utilisés peuvent être propres à une version du système.

- **Conception**

On y modélise tous les rouages d'implémentation et on détaille tous les éléments de modélisation issus des niveaux supérieurs.
Les modèles sont optimisés, car destinés à être implémentés.

1.7 Les diagrammes

1.7.1 Définition d'un diagramme

Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle.
C'est une perspective du modèle, pas "le modèle".
Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis).
Un type de diagramme UML véhicule une sémantique précise (un type de diagramme offre toujours la même vue d'un système).
Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.
Par extension et abus de langage, un diagramme UML est aussi un modèle (un diagramme modélise un aspect du modèle global).

UML 1.3 propose 9 diagrammes tandis qu'UML 2 en propose 13.

Il existe 2 types de vues du système qui comportent chacune leurs propres diagrammes :

- les vues statiques :
  - diagrammes de cas d'utilisation
  - diagrammes d'objets
  - diagrammes de classes
  - diagrammes de composants
  - diagrammes de déploiement
  - diagramme des paquetages
  - diagramme de structure composite (UML2)

- les vues dynamiques :
  - diagrammes de collaboration
  - diagrammes de séquence / Diagramme de communication (UML2)
  - diagrammes d'états-transitions
  - diagrammes d'activités
  - diagramme global d'interaction (UML2)
  - diagramme de temps (UML2)

1.7.2 Vues statique du système

Le but de la conceptualisation est de comprendre et structurer les besoins du client. : Il ne faut pas chercher l'exhaustivité, mais clarifier, filtrer et organiser les besoins.
Une fois identifiés et structurés, ces besoins :
- définissent le contour du système à modéliser (ils précisent le but à atteindre),
- permettent d'identifier les fonctionnalités principales (critiques) du système.

Le modèle conceptuel doit permettre une meilleure compréhension du système.
Le modèle conceptuel doit servir d'interface entre tous les acteurs du projet.
Les besoins des clients sont des éléments de traçabilité dans un processus intégrant UML.
Le modèle conceptuel joue un rôle central, il est capital de bien le définir.

1.7.2.1 Diagrammes de cas d'utilisation

a) DEFINITION DU CAS D'UTILISATION (USE CASE)

Les use cases permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système. Ils centrent l'expression des exigences du système sur ses utilisateurs : ils partent du principe que les objectifs du système sont tous motivés.
La détermination et la compréhension des besoins sont souvent difficiles car les intervenants sont noyés sous de trop grandes quantités d'informations : il faut clarifier et organiser les besoins des clients (les modéliser). Pour cela, les cas d'utilisation identifient les utilisateurs du système (acteurs) et leurs interactions avec le système. Ils permettent de classer les acteurs et structurer les objectifs du système.
Une fois identifiés et structurés, ces besoins :
- définissent le contour du système à modéliser (ils précisent le but à atteindre),
- permettent d'identifier les fonctionnalités principales (critiques) du système.

Les use cases ne doivent donc en aucun cas décrire des solutions d'implémentation.
Leur but est justement d'éviter de tomber dans la dérive d'une approche fonctionnelle, où l'on liste une litanie de fonctions que le système doit réaliser.

b) ELEMENTS DE MODELISATION DES CAS D'UTILISATION

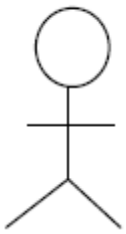
- **L'acteur** :

La première étape de modélisation consiste à définir le périmètre du système, à définir le contour de l'organisation et à le modéliser. Toute entité qui est en dehors de cette organisation et qui interagit avec elle est appelé acteur selon UML.
Un acteur est un type stéréotypé représentant une abstraction qui réside juste en dehors du système à modéliser.
Un acteur représente un rôle joué par une personne ou une chose qui interagit avec le système (la même personne physique peut donc être représentée par plusieurs acteurs en fonction des rôles qu'elle joue).
Pour identifier les acteurs, il faut donc se concentrer sur les rôles joués par les entités extérieures au périmètre.
Dans UML, il n'y a pas de notion d'acteur interne et externe. Par définition, un acteur est externe au périmètre de l'étude, qu'il appartienne ou pas à la société.
Enfin, un acteur n'est pas nécessairement une personne physique : il peut être un service, une société, un système informatique …

Il existe 4 catégories d'acteurs :

- les acteurs principaux : les personnes qui utilisent les fonctions principales du système
- les acteurs secondaires : les personnes qui effectuent des tâches administratives ou de maintenance.
- le matériel externe : les dispositifs matériels incontournables qui font partie du domaine de l'application et qui doivent être utilisés.
- les autres systèmes : les systèmes avec lesquels le système doit interagir.

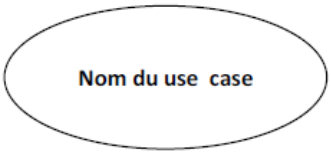
Formalisme :



Nom acteur

- **Le cas d'utilisation** :

Le cas d'utilisation (ou use case) correspond à un objectif du système, motivé par un besoin d'un ou plusieurs acteurs.
L'ensemble des use cases décrit les objectifs (le but) du système.

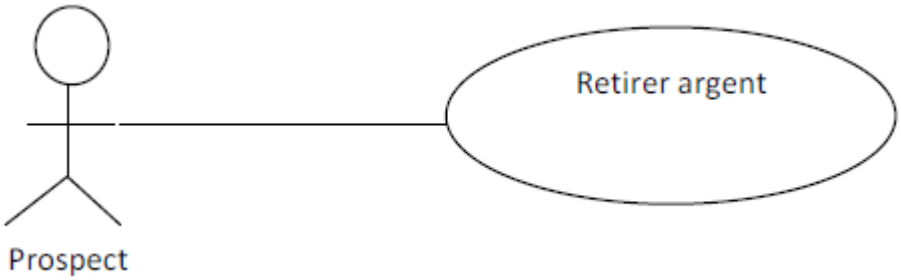


Formalisme :

- **La relation**

Elle exprime l'interaction existant entre un acteur et un cas d'utilisation.

Formalisme :

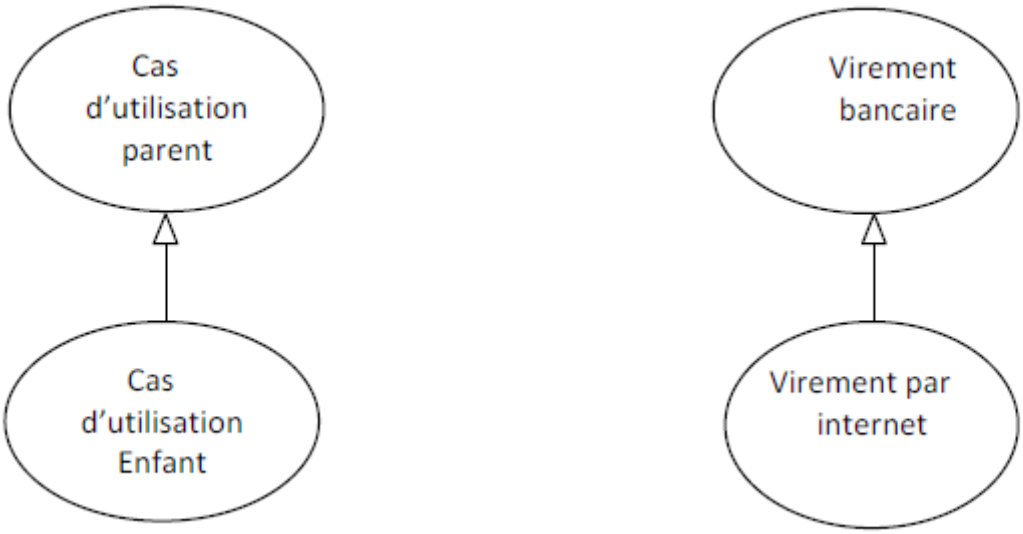


Il existe 3 types de relations entre cas d'utilisation :
- la relation de généralisation
- la relation d'extension
- la relation d'inclusion

- **La relation de généralisation**

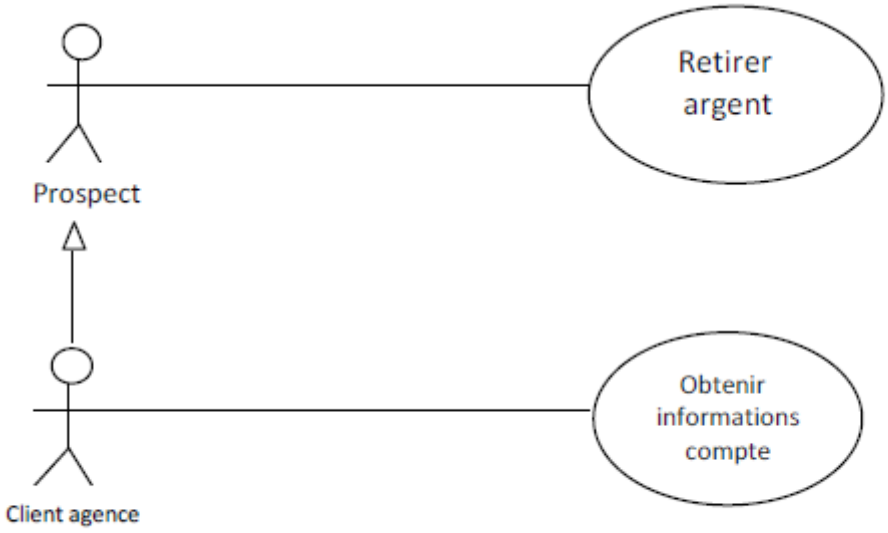
Dans une relation de généralisation entre 2 cas d'utilisation, le cas d'utilisation enfant est une spécialisation du cas d'utilisation parent.

Formalisme et exemple :



NB : un acteur peut également participer à des relations de généralisation avec d'autres acteurs. Les acteurs « enfant » seront alors capables de communiquer avec les mêmes cas d'utilisation que les acteurs « parents ».

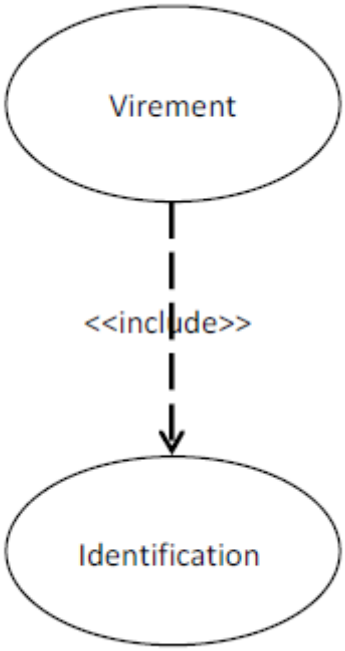
Formalise exemple :



- **La relation d'inclusion**

Elle indique que le cas d'utilisation source contient aussi le comportement décrit dans le cas d'utilisation destination. L'inclusion a un caractère obligatoire, la source spécifiant à quel endroit le cas d'utilisation cible doit être inclus. Cette relation permet ainsi de décomposer des comportements et de définir des comportements partageables entre plusieurs cas d'utilisation.

Formalise :

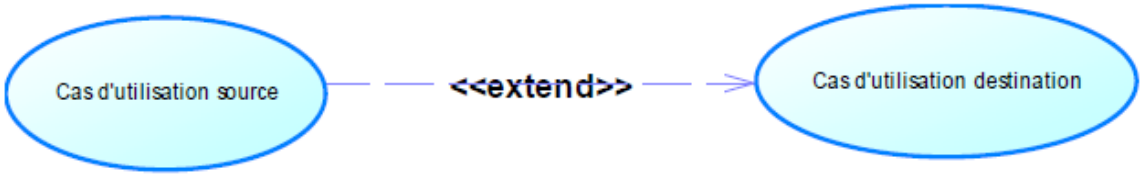


Pour réaliser l'objectif « virement », on utilise obligatoirement « identification

- **La relation d'extension**

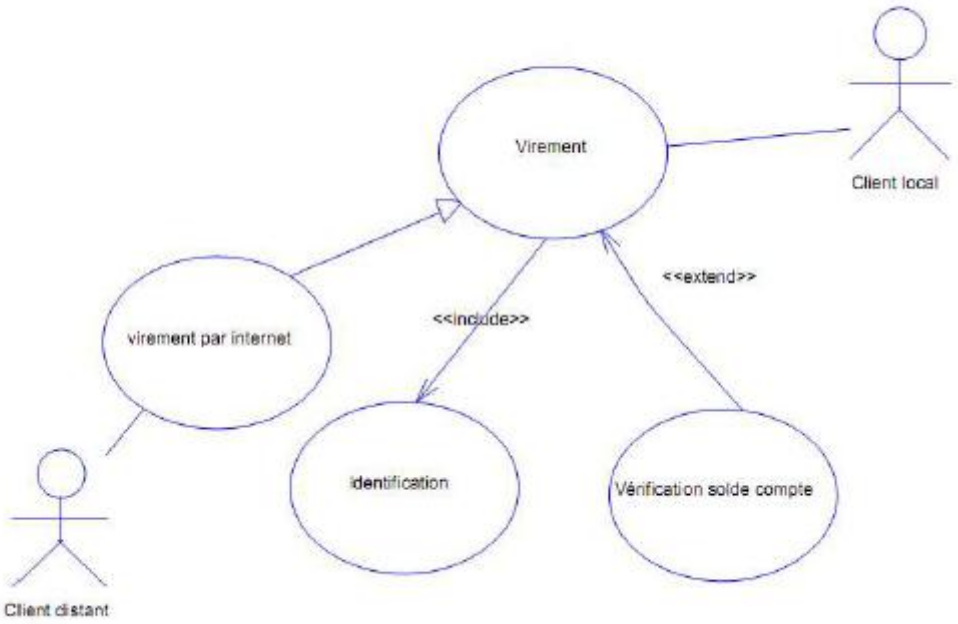
Elle indique que le cas d'utilisation source ajoute son comportement au cas d'utilisation destination. L'extension peut être soumise à condition. Le comportement ajouté est inséré au niveau d'un point d'extension défini dans le cas d'utilisation destination. Cette relation permet de modéliser les variantes de comportement d'un cas d'utilisation (selon les interactions des acteurs et l'environnement du système).

Formalisme :



- **Paquetage**

Un paquetage (package) est un groupement d'élément de modélisation. Un paquetage peut contenir aussi bien des paquetages emboîtés que des éléments de modélisation ordinaires.
Le système entier peut être pensé comme un unique paquetage de haut niveau comprenant l'ensemble. Tous les éléments de modélisation d'UML, y compris les diagrammes, peuvent être organisés en paquetage.
Les uses cases peuvent être organisés en paquetages (packages).



- **Les scénarios**

Un cas d'utilisation est une abstraction de plusieurs chemins d'exécution. Une instance de cas d'utilisation est appelée : « scénario ».

Chaque fois qu'une instance d'un acteur déclenche un cas d'utilisation, un scénario est créé (le cas d'utilisation est instancié). Ce scénario suivra un chemin particulier dans le cas d'utilisation.

Un scénario ne contient pas de branche du type « Si condition … alors » car pendant l'exécution, la condition est soit vraie, soit fausse, mais elle aura une valeur.
Après la description des cas d'utilisation, il est nécessaire de sélectionner un ensemble de scénarios qui vont servir à piloter l'itération en cours de développement.
Le choix et le nombre de scénarios à retenir est une étape difficile à réaliser : l'exhaustivité est difficile, voire impossible à atteindre. Le nombre d'instances pour un cas d'utilisation peut être très important, voire infini.
Les scénarios peuvent être classés en :

- scénarios principaux : il correspond à l'instance principale du cas d'utilisation.

C'est souvent le chemin « normal » d'exécution du cas d'utilisation qui n'implique pas d'erreurs.

- Scénarios secondaires : il peut être un cas alternatif (un choix), un cas exceptionnel ou une erreur.

Les scénarios sont utiles pour :

- analyser et concevoir le système
- justifier les choix effectués (ils serviront à la documentation des cas d'utilisation)
- tester : les scénarios constituent le meilleur moyen de spécifier les tests.

1.7.2.2 Diagrammes de classes

a) DEFINITION DU DIAGRAMME DE CLASSES

Le diagramme de classes exprime la structure statique du système en termes de classes et de relations entre ces classes.
L'intérêt du diagramme de classe est de modéliser les entités du système d'information.
Le diagramme de classe permet de représenter l'ensemble des informations finalisées qui sont gérées par le domaine. Ces informations sont structurées, c'est-à-dire qu'elles ont regroupées dans des classes. Le diagramme met en évidence d'éventuelles relations entre ces classes.

Le diagramme de classes comporte 6 concepts :

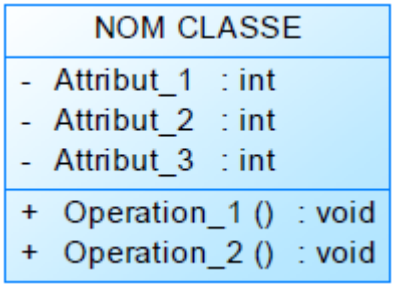
- classe
- attribut
- identifiant
- relation
- opération
- généralisation / spécialisation

b) LES NOTIONS UTILISEES PAR LE DIAGRAMME DE CLASSES

- **La notion de classe**

Définition : une classe est une description abstraite (condensée) d'un ensemble d'objets du domaine de l'application : elle définit leur structure, leur comportement et leurs relations.
Représentation : les classes sont représentées par des rectangles compartimentés :

- le 1er compartiment représente le nom de la classe
- le 2ème compartiment représente les attributs de la classe
- le 3ème compartiment représente les opérations de la classe



Les compartiments d'une classe peuvent être supprimés (en totalité ou en partie) lorsque leur contenu n'est pas pertinent dans le contexte d'un diagramme. La suppression des compartiments reste purement visuelle : elle ne signifie pas qu'il n'y a pas d'attribut ou d'opération.
Le rectangle qui symbolise une classe peut contenir un stéréotype et des propriétés.
UML définit les stéréotypes de classe suivants :
- « classe implémentation » : il s'agit de l'implémentation d'une classe dans un langage de programmation
- « énumération » : il s'agit d'une classe qui définit un ensemble d'identificateurs formant le domaine de la valeur.
- « métaclasse » : il s'agit de la classe d'une classe, comme en Smalltalk
- « powertype » : une classe est un métatype : ses instances sont toutes des sous-types d'un type donné
- « processus » : il s'agit d'une classe active qui représente un flot de contrôles lourd
- « thread » : il s'agit d'une classe active qui représente un flot de contrôles léger
- « type » : il s'agit d'une classe qui définit un domaine d'objets et les opérations applicables à ces objets.
- « utilitaire » : il s'agit d'une classe réduite au concept de module et qui ne peut être instanciée.

■ La notion d'attribut

Définition : Une classe correspond à un concept global d'information et se compose d'un ensemble d'informations élémentaires, appelées attributs de classe.

Un attribut représente la modélisation d'une information élémentaire représentée par son nom et son format.
Par commodité de gestion, on choisit parfois de conserver dans un attribut le résultat d'un calcul effectué à partir d'autres classes : on parle alors d'attribut dérivé. Pour repérer un attribut dérivé : on place un / devant son nom.

Visibilité et portée des attributs :
UML définit 3 niveaux de visibilité pour les attributs :
1- public (+) : l'élément est visible pour tous les clients de la classe
2- protégé (#) : l'élément est visible pour les sous-classes de la classe
3- privé (-) : l'élément n'est visible que par les objets de la classe dans laquelle il est déclaré.

■ La notion d'identifiant

L'identifiant est un attribut particulier, qui permet de repérer de façon unique chaque objet, instance de la classe.

■ La notion d'opération

Définition : l'opération représente un élément de comportement des objets, défini de manière globale dans la classe.

Une opération est une fonctionnalité assurée par une classe. La description des opérations peut préciser les paramètres d'entrée et de sortie ainsi que les actions élémentaires à exécuter.
Visibilité et portée des opérations :
Comme pour les attributs, on retrouve 3 niveaux de visibilité pour les opérations :

1- public (+) : l'opération est visible pour tous les clients de la classe
2- protégé (#) : l'opération est visible pour les sous-classes de la classe
3- privé (-) : l'opération n'est visible que par les objets de la classe dans laquelle elle est déclarée.

- **La notion de relation**

S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.
Les liens entre les objets doivent être considérés comme des instances de relations entre classes.
Il existe plusieurs types de relations entre classes :

- l'association
- la généralisation/spécialisation
- la dépendance

L'association

L'association est la relation la plus courante et la plus riche du point de vue sémantique.
Une association est une relation statique n-aire (le plus souvent : elle est binaire) : c'est-à-dire qu'elle relie plusieurs classes entre elles.
L'association existe entre les classes et non entre les instances : elle est introduite pour montrer une structure et non pour montrer des échanges de données.
Une association n-aire possède n rôles qui sont les points terminaux de l'association ou terminaisons.
Chaque classe qui participe à l'association joue un rôle. Les rôles sont définis par 2 propriétés :

- la multiplicité : elle définit le nombre d'instances de l'association pour une instance de la classe. La multiplicité est définie par un nombre entier ou un intervalle de valeurs. La multiplicité est notée sur le rôle (elle est notée à l'envers de la notation MERISE).

1	Un et un seul
0..1	Zéro ou un
N ou *	N (entier naturel)
M..N	De M à N (entiers naturels)
0..*	De zéros à plusieurs
1..*	De 1 à plusieurs

Les valeurs de multiplicité expriment les contraintes liées au domaine de l'application. Il est donc important de déterminer les valeurs de multiplicité optimales pour trouver le bon équilibre entre complexité et efficacité. La surestimation des valeurs de multiplicité entraîne un surcoût de taille de stockage et en vitesse d'exécution (requête avec plus de jointures).

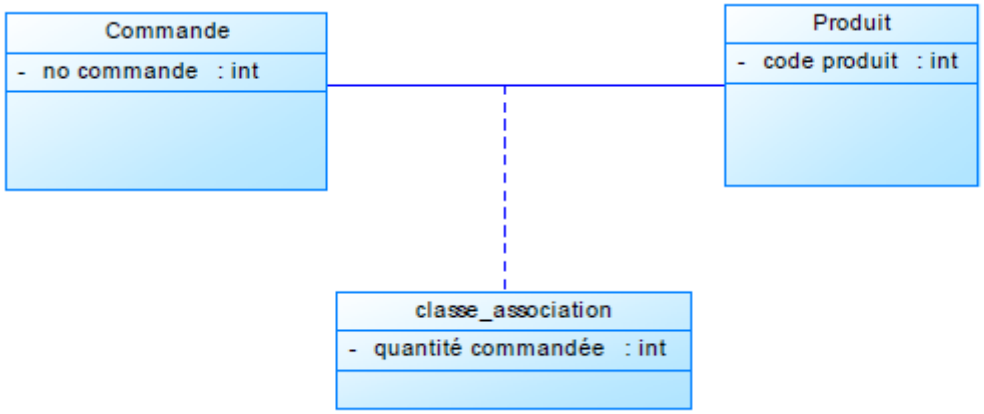
- **la navigabilité**

La navigabilité n'a rien à voir avec le sens de lecture de l'association. Une navigabilité placée sur une terminaison cible indique si ce rôle est accessible à partir de la source.
Par défaut les associations sont navigables dans les 2 sens. Dans certains cas, une seule direction de navigation est utile : l'extrémité d'association vers laquelle la navigation est possible porte alors une flèche.

- **Les classes-association**

Les attributs d'une classe dépendent fonctionnellement de l'identifiant de la classe. Parfois, un attribut dépend fonctionnellement de 2 identifiants, appartenant à 2 classes différentes.

Par exemple, l'attribut « quantité commandée » dépend fonctionnellement du numéro de commande et du code produit. On va donc placer l'attribut « quantité commandée » dans l'association « comporter ».
Dans ce cas, l'association est dite « porteuse d'attributs ».
Une association porteuse d'attributs est appelée classe-association.

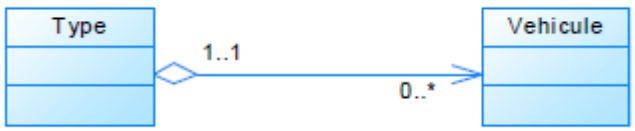


- **L'agrégation**

Dans UML, l'agrégation n'est pas un type de relation mais une variante de l'association.
Une agrégation représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité.
L'agrégation ne peut concerner qu'un seul rôle d'une association.
L'agrégation se représente toujours avec un petit losange du côté de l'agrégat.
Le choix d'une association de type agrégation traduit la volonté de renforcer la dépendance entre classes. C'est donc un type d'association qui exprime un couplage plus fort entre les classes.
L'agrégation permet de modéliser des relations de type maître et esclaves.
L'agrégation permet de modéliser une contrainte d'intégrité et de désigner l'agrégat comme contrainte.
A travers une telle contrainte, il est possible de représenter par exemple :

- la propagation des valeurs d'attributs d'une classe vers une autre classe
- une action sur une classe qui implique une action sur une autre classe
- une subordination des objets d'une classe à une autre classe

Formalisme et exemple :

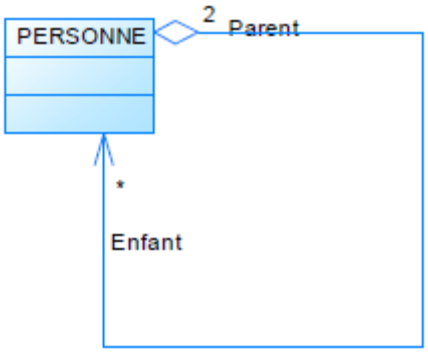


L'exemple ci-dessus montre que l'on veut gérer une classification de véhicules. Chaque véhicule est classifié selon son type. En conséquence, il sera possible de prendre connaissance pour un véhicule de l'ensemble des caractéristiques du type de véhicule auquel il est associé.

NB : un agrégat peut être multiple. Dans l'exemple ci-dessous, un véhicule peut appartenir à plusieurs types.

Cas particulier des associations réflexives :

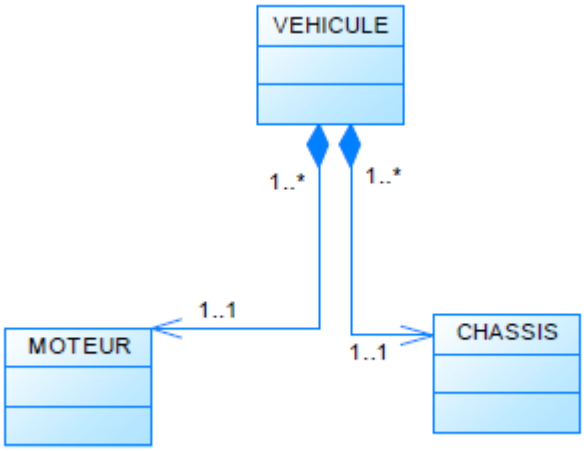
On peut avoir des cas d'agrégation réflexive dès que l'on modélise des relations hiérarchiques ou des liens de parenté par exemple.



- **La composition**

La composition est un cas particulier de l'agrégation dans laquelle la vie des composants est liée à celle des agrégats. Elle fait souvent référence à une contenance physique. Dans la composition l'agrégat ne peut être multiple.
La composition implique, en plus de l'agrégation, une coïncidence des durées de vie des composants : la destruction de l'agrégat (ou conteneur) implique automatiquement la destruction de tous les composants liés.

Formalisme et exemple :



Partie 4 : Biographie

Sites web :

APEC.fr

https://agiliste.fr

Livres :

- Kenneth Laudon et Jane Laudon, « Management des systèmes d'information», 2017, Pearson
- Jean-François PILLOU et Pascal CAILLEREZ, « Tout sur les Système d'information », 3e édition, 2016, DUNOD
- Andrew Tanenbaum, « Réseaux », 5e édition, 2011, PERASON
- Pujolle, « Les réseaux », 9e édition, EYROLLES
- Jean-François PILLOU et Pascal CAILLEREZ, « Tout sur les Réseaux et Internet », 4e édition, 2015, DUNOD
- Guillaume PLOUIN, « Cloud Computing », 4e édition, 2016, DUNOD
- Tom Marrs, « Json at Work », 2017, O'Reilly
- Jacques Lonchamp, « Analyse des besoins pour le développement logiciel », 2015, DUNOD
- Bassem EL HADDAD et Julien OGER De la théorie à la pratique, 2e édition, 2019, EYROLLES
- Bassem EL HADDAD et Julien OGER, Mémento Scrum

Tutoriels :

- Méthode RAD-Elements fondamentaux, de Jean-Pierre Vickoff

Cours :

- Lean Software Development de Pablo Pernot, 2014.
- L'essentiel de XML, cours XML d'Olivier Carton, 2015
- UML 2.0 de Laurent AUDIBERT