Architectures de Systèmes d'Information

Livre Blanc

Novembre 2002

Table des matières

PREFACE de François Tabourot **2**

Comment l'architecture et l'urbanisme apportent une réponse aux nouveaux enjeux de rentabilité de l'entreprise **3**

I. Introduction **7**

La « révolution Internet » aura-t-elle lieu ? **7**

Des conséquences inattendues sur le marché informatique… **8**

Restaurer la confiance nécessitera d'augmenter le taux de réussite des projets **9**

Pourquoi ne pas appliquer à l'informatique les méthodes éprouvées dans d'autres secteurs ? **10**

La technologie n'est pas un messie, elle fait simplement partie de l'histoire …..... **11**

La gestion du changement s'appliquera aussi aux DSI… **12**

II. Démarche d'architecture d'entreprise **13**

Mille et une DSI **13**

A quoi sert une DSI ? **14**

La DSI, une SSII comme les autres ? **15**

Conclusion **17**

III. Etat de l'art des principes d'architecture **18**

L'Architecture Logicielle **18**

L'Architecture de SI **24**

IV. Les outils de l'architecte **33**

Le formalisme de modélisation **33**

Des patterns pour structurer la conception **40**

V. Démarche d'architecture projet **48**

L'Etude Stratégique **50**

L'Etude d'Architecture Générale **52**

L'Etude d'Architecture Détaillée **69**

Le bureau d'architecture en phase d'implémentation **75**

VI. Bibliographie **80**

VII. A propos d'OCTO Technology **81**

Préface de François Tabourot

Directeur Général MEGA International

Comment l'architecture et l'urbanisme apportent une réponse aux nouveaux enjeux de rentabilité de l'entreprise

Tout en changeant de nature et d'intensité, la demande de justification économique des dépenses se fait de plus en plus pesante sur les directeurs informatiques.

C'est pour répondre à cette transformation des attentes du management que les outils de pilotage dont les directions informatiques ont besoin évoluent.

C'est par cet angle que nous proposons de comprendre le fondement des approches de cartographie et d'urbanisme de systèmes d'information.
Comme l'architecte et l'urbaniste œuvrent au développement de l'habitat et des cités, le directeur informatique dispose avec ces nouveaux outils d'un ensemble de cartes et de représentations utiles pour analyser, construire et adapter les systèmes d'information aux grands enjeux métier de l'entreprise. Mais ces outils doivent aussi s'inscrire dans une démarche de rentabilité, comme nous allons le montrer dans cette préface.

De l'organisation des ressources à la justification économique

A l'origine, il n'est pas question de justification économique : lorsque la modernisation entraîne avec elle, la promesse du dynamisme de la croissance et du progrès social, quel décideur économique voudrait s'y opposer ?
C'est ainsi que se font les grands progrès, par rupture, lorsque les évidences et les fantasmes initiaux sont assez forts pour atténuer la visibilité des difficultés qu'imposera le changement.
C'est de cette façon que s'est développée l'informatique de gestion dans l'entreprise à la fin des années 70 : pouvait-on imaginer ne pas s'informatiser ? Nier que l'apport fonctionnel de cette technologie était un levier de modernité au bénéfice de l'homme et de son rapport au travail ?
Aucune entreprise n'a contesté le besoin d'informatisation, pour rester dans les standards de compétitivité. L'amélioration de la productivité par l'automatisation des tâches administratives se suffisait à elle-même, comme évidence initiale pour déclencher un vaste mouvement d'informatisation.
Sous la responsabilité de la direction administrative et financière, une nouvelle direction, la direction informatique, s'est alors vue chargée de faire l'acquisition et de déployer dans les meilleures conditions ces nouvelles technologies : machine, base de donnée, langage, système d'exploitation…
Ensuite, l'évolution technologique s'inscrivant sans rupture dans la continuité de la révolution fonctionnelle initiale, le besoin de justification économique est apparu et s'est intensifié avec le temps.
Nous allons montrer comment l'urbanisme contribue à la rentabilité de l'entreprise, d'abord de manière simple, en permettant d'optimiser le système d'information par rapport aux activités de l'entreprise, mais aussi de manière innovante, en permettant d'améliorer sa contribution à la production de valeur dans l'entreprise.

Urbanisme et réduction des coûts

Dès lors que l'impact d'une innovation sur le « business model » n'est pas évident, tout choix technologique doit être justifié par le retour sur investissement. C'est à dire que le directeur informatique doit prouver que le même service peut fondamentalement être rendu à moindre coût : d'exploitation, de maintenance, de déploiement…
Le cahier des charges fixé au directeur informatique par le management l'oriente alors vers la recherche d'économies, sans dégradation du service rendu par l'informatique à l'entreprise. Et le rôle du directeur informatique consiste à sélectionner les solutions les plus adaptées parmi les offres des fournisseurs, selon trois axes de réduction de coûts possibles :

- Diminution des coût d'exploitation des architectures matérielles par migration des technologies « gros systèmes » ;
- Diminution du coût des développements par utilisation des progiciels et - généralisation des principes de développement basés sur la réutilisation ;
- Rationalisation du parc applicatif par apurement du « legacy system ».

Pour faire face à une offre foisonnante et fortement combinatoire sur le plan des solutions technologiques (réseau, base de données, machines…), les directions informatiques ont ainsi été amenées à mettre en place une responsabilité « d'architecture technique ». Cette structure, souvent renforcée d'une expertise externe, assume la responsabilité du « down-sizing » des coûts informatiques par le levier de l'infrastructure, en adressant des enjeux de diminution des coûts d'exploitation et de mutualisation des ressources.
Sur l'axe applicatif, et sensiblement au même moment, comment nier l'impact considérable de l'arrivée des progiciels sur l'évolution des responsabilités et du mode de raisonnement du directeur ou responsable des études ? Fabriquer du code s'inscrit de la même manière dans une justification économique drastique où l'on doit démontrer que le même service n'existe pas déjà dans le parc applicatif, ni à l'extérieur sous forme d'un progiciel. Il est désormais admis que faire coûte toujours plus cher qu'acheter tout fait (ici l'histoire viendra pondérer ce point de vue, en particulier sur la base du retour d'expérience du déploiement des ERP).
Si l'on choisit de développer en interne, on est contraint de démontrer que tant le choix du langage que celui de l'architecture fonctionnelle minimise les coûts de développement et maximise la réutilisation des composants.
Dans ce contexte, et pour répondre à cette pression économique basée sur la réduction des coûts, les directeurs informatiques ont besoin de nouveaux outils de pilotage de leur activité et d'instrumentation de leur relation avec le management.
La cartographie urbanisée des systèmes d'information apparaît comme le moyen pour le directeur informatique de garantir l'optimum de l'utilisation de ses ressources matérielles et applicatives.
Elle lui permet en effet :

1. de parfaitement connaître et maîtriser l'ensemble de ses ressources ;
2. de comprendre leur niveau d'interdépendance et de réutilisabilité ;
3. de garantir la facilité d'accès à cette information documentaire, en rangeant l'ensemble de ces éléments à l'intérieur de repères proches de l'activité de l'entreprise.

Cette nomenclature statique d'activités constitue le cadre de référence métier permettant d'urbaniser l'ensemble des ressources.
Cette cartographie est un projet délimité essentiellement au sein du service informatique. Elle n'implique pas de connaître le fonctionnement de l'entreprise, les grands processus métier, la collaboration des activités et des acteurs. Beaucoup de projets de cartographie sont ainsi menés sans grande implication des directions fonctionnelles.
Selon cette approche, la cartographie est essentiellement un outil d'optimisation utile à la direction informatique.

En décrivant les différents composants du système d'information, leurs relations et leur utilisation pour servir les différents métiers de l'entreprise, on dispose en effet d'un outil d'optimisation complet pour :

- analyser les redondances et repérer les applications inutilisées ;
- mesurer l'impact d'une opération de maintenance et rationaliser sa mise en œuvre : ce n'est pas pour rien que ces approches ont connu leur heure de gloire lors des projets an 2000 ;
- faciliter les projets d'acquisition de nouvelles technologies, en particulier d'intéropérabilité de systèmes, workflow et EAI.

Ainsi, comme il s'attache à une simple lecture des systèmes d'information et de leur position sur le planisphère de l'organisation de l'entreprise, l'urbanisme des systèmes d'information dans son acception courante (théorie de Jacques Sassoon), et par extrapôlation des théories de l'urbanisme traditionnel, accompagne la démarche de réduction des coûts, dont l'objectif principal est la recherche de l'optimum.
Cette analyse des outils d'urbanisme met en évidence leur propre limite comme tous les outils d'optimisation. Pour prendre le langage des financiers, on comprend que la fabrication de marge par réduction de dépenses n'est pas un facteur d'accroissement du chiffre d'affaires.
Qu'autrement dit et très naturellement on comprend qu'ayant démontré leur capacité à servir le fonctionnement du « business model » de l'entreprise au moindre coût, la question émergente posée aux directions informatique est d'envisager leur capacité à contribuer significativement à la production de valeur.

C'est à dire à proposer des investissements informatiques (matériels ou applicatifs) dont le retour sur investissement se calculera sur la base d'un gain de productivité d'un processus métier, de la conquête de parts de marché, de la capacité à se diversifier, donc sur la base d'une vue dynamique du fonctionnement de l'entreprise.

Si l'on assume qu'urbaniser c'est positionner des objets sur des repères, ce nouvel urbanisme naissant est celui qui permet d'envisager la contribution des ressources informatiques (les objets) à la performance des chaînes de valeur de l'entreprise (les repères).

Urbanisme et production de valeur

Il s'agit donc désormais de démontrer comment l'informatique permet, non seulement de rentabiliser ses investissements propres, mais aussi de produire de la valeur pour l'entreprise.
Au delà de la réduction des coûts dans l'informatique, l'enjeu pour la direction informatique concerne en effet sa capacité à contribuer de manière directe à la production de valeur.
Cela signifie par exemple :

- utiliser les outils de « business intelligence » pour mieux qualifier les contacts lors d'une prise de commande, et par suite diminuer les taux d'impayés et augmenter le panier moyen en VPC ;

- organiser des échanges automatisés et intégrer le système de gestion de livraison d'un sous-traitant pour proposer en ligne à ses clients les meilleures solutions de résolution de panne ;

- développer une gestion automatisée des processus pour organiser le back office de l'ensemble de ses agences, et assurer ainsi une réponse centralisée à tous les dossiers de prêts transmis par les agents dans un délai garanti.

L'urbanisme « nouveau » permet de rapprocher la cartographie du système d'information de la vision dynamique des chaînes de valeur, c'est à dire des collaborations entre activités et acteurs pour produire de la valeur. Le projet d'urbanisme implique alors les grandes directions métier de l'entreprise, pour comprendre comment l'informatique contribue à la stratégie de l'entreprise.
Dans cette démarche, le directeur informatique se rapproche d'une fonction de « chef de progrès » décrite à la fin des années 80 par André Leynaud[1] dans ses traités sur l'informatique stratégique.
Tel est le challenge de l'entreprise d'aujourd'hui : penser son positionnement concurrentiel et sa création de valeur sur le marché, en corrélation directe avec l'architecture de ses ressources en particulier informatiques.
C'est la nouvelle problématique d'urbanisme de systèmes d'information, qui recentre le directeur informatique sur les enjeux métier de l'entreprise, au risque d'une moins grande maîtrise de la dimension technologique des systèmes d'information, éventuellement sous-traitée à des tiers.
L'entreprise fera ainsi de son responsable informatique un « business architecte », capable de prendre en main la déclinaison opérationnelle de la stratégie.
Pour s'y préparer il lui faut d'ores et déjà des outils de pilotage constitués en fond documentaire qui précise :

- Les chaînes de valeur, leur description, et les performances stratégiques attendues ;
- L'architecture des ressources, et en particulier, leur contribution à ces chaînes de valeur.

C'est la vue business du système d'information urbanisé selon les processus métier de l'entreprise.

Un nouveau rôle pour le Directeur Informatique

Si le cycle complet de mise en place d'un système d'information obéit toujours à la même problématique, l'intensification de la problématique associée à chaque étape justifie selon nous une spécialisation et un partage des rôles.
Ainsi, c'est souvent l'intégrateur extérieur qui, assumant la responsabilité de la « performance informatique », aura l'initiative de la technologie, alors que le directeur informatique rendra compte au comité de direction de l'atteinte d'objectifs business.

1. Quand le service est identifié et que le problème se limite à en optimiser le coût, alors le métier informatique devient un métier d'industriel, seul capable :
   - de maîtriser le foisonnement technologique et faire émerger des formes de standardisation ;
   - de générer des économies par mutualisation des coûts.
2. Quand les services rendus par la technologie sont des leviers de transformation du « business model », de son efficacité et de sa compétitivité, les hommes et les femmes qui pensent l'architecture fonctionnelle du système d'information pilotent la réflexion stratégique sur la base de leur double maîtrise :
   - des mécanismes de création de valeur, les « business process » ;
   - de la contribution des services fournis à ces « business process ».

   S'ils sont les maitres d'ouvrage du système d'information, ils doivent l'être d'avantage pour leur connaissance du fonctionnement et des hommes de l'entreprise, que pour celle des règles de gestion contenues dans les systèmes.

1 : André Leynaud fut le Président du Gamma International au sein du Groupe Hay, puis de Cap Gemini.

C'est, selon notre analyse l'une des causes principales du revers de « la nouvelle économie » qui, en privilégiant l'approche strictement technologique, a occulté la lecture de certaines transformations de business model pourtant très significatives et portant sur les modes de consommation, d'achats, de distribution…
De la même manière, mais sur le plan de la micro économie, la mise en place des grands ERP atteint finalement son retour sur investissement lorsque enfin, quelqu'un s'intéresse à comprendre comment les utilisateurs vont travailler mieux avec de nouveaux outils, indépendamment de la manière dont ils sont fabriqués.

Maîtrise des processus métiers, de l'architecture corrélative des hommes et des ressources qui contribuent à leur fonctionnement et leur performance, est plus que jamais une attente forte dans une macro économie qui accélère considérablement le besoin de flexibilité du « busines model ».

Est-il donc nécessaire de convaincre le décideur économique du bien fondé de l'urbanisme des systèmes d'information, lorsque, au bout du compte, on démontre que c'est lui qui en formule la demande ?
Sans doute que le jargon informatique encore trop souvent employé pour décrire l'urbanisme ne sert pas le directeur informatique. Il lui appartient aussi d'en faire un outil de communication pour son management, pour prendre sa place véritable au sein des directions d'entreprise.

I. Introduction

La « révolution Internet » aura-t-elle lieu ?

Septembre 2000. Les espoirs de gains de productivité liés à la mise en place des Nouvelles Technologies de l'Information et de la Communication (NTIC) dans les entreprises sont au plus haut. Ils justifient la démesure des valorisations de toute société ayant plus ou moins vaguement affiché des e-ambitions. Malheureuse, ringarde, celle qui n'est pas une e-entreprise, qui n'a pas son projet eProcurement[2], eCRM[3], eSCM[4], ou qui n'offre pas l'accès à son catalogue de tôles d'acier laminé sur le WAP.

Les promesses de nombreux chantiers informatiques ces cinq dernières années n'ont pas été tenues.

Septembre 2002. Le « e » est passé de mode, ensemble avec les trotinettes et les stocks-options, et s'est définitivement paré d'un voile de désuétude comme seule la disparition des pantalons à pattes d'éléphant et des sous-pulls en acrylique nous avait habitués : pouah, plus jamais ça.

Internet devait accélérer la marche du progrès, on n'aurait jamais imaginé à quel point cette vitesse imaginaire consumerait sa crédibilité au point de lui brûler les ailes. Car tel est le sens de la crise, une crise de confiance, une sorte de désillusion face à une supposée révolution, ou nouvelle économie, qui n'a laissé la place qu'à des ardoises faramineuses en lieu et place des économies et accroissements de la compétitivité attendus.

Or les gains de productivité liés à une démarche d'informatisation de processus, quels qu'ils soient (marketing, vente, approvisionnement, achat, etc.) apparaissent conditionnés à la dématérialisation globale dudit processus et à une réorganisation du travail autour de ce processus.

On oublie trop souvent que les technologies de l'information sont avant tout des technologies au service de l'organisation.

Ceci a deux conséquences, l'une informatique pour laquelle la complexité et l'étendue des chantiers ont été sous-estimées. L'autre, organisationnelle, renvoie au fameux paradoxe de Solow[5], qui ne voyait dans l'augmentation de l'utilisation de l'informatique aucune augmentation de la productivité (« l'informatique se voit partout, sauf dans les statistiques »). Or ce paradoxe n'en est plus un lorsqu'on observe les entreprises qui accompagnent leurs projets informatiques des réorganisations nécessaires à la présence de ce nouveau « personnel numérique ».

Ainsi, la plupart des projets CRM mis en place pour des fonctions ciblées ont atteint leurs objectifs : centre d'appels, infocentre commerciaux (churn[6], rentabilité produit…), gestion de contacts, etc. Mais les projets himalayens ont souvent accouché de souris, quand ils ont accouché. Et pour cause, on avait simplement oublié d'imaginer que c'est tout l'univers de l'entreprise qui serait bouleversé par la mise en place de tels systèmes : reprise de contrôle en central, regroupement de compétences, modification des centres de profit, nouveaux services à vendre au client, etc.

L'entreprise est un tout, l'informatique ne doit pas être vue comme la clé de voûte unique du changement. Si la production n'est pas flexible, inutile de s'équiper d'un système complexe de gestion de la chaîne logistique ! L'innovation technique doit aller de pair avec des innovations organisationnelles : 'développement du « juste à temps », et du « sur-mesure » avec externalisation massive'[7].

2 : Gestion électronique des achats.
3 : Gestion électronique de la relation client.
4 : Gestion électronique de la chaîne d'approvisionnement.
5 : Economiste américain, prix Nobel d'économie en 1987.
6 : Taux de Churn : facteur qui mesure la propension d'un client à changer de fournisseur, très utilisé dans le secteur des télécommunications.
7 : Nouvelle économie : rapport Daniel Cohen et Michèle Debonneuil.

L'informatisation doit accompagner les changements d'organisation à l'intérieur de l'entreprise ; elle peut aussi les susciter.

A ce registre, les meilleurs exemples susceptibles d'étayer cette thèse se trouvent auprès des entreprises « virtuelles », qui se sont construites précisément dans des modèles modulaires devant pallier l'absence de ciment capitalistique. Ici l'outil informatique est un facteur clé d'intégration d'acteurs indépendants dans une structure cohérente.

Les entreprises virtuelles qui ont connu les succès les plus durables, comme Ikea, Benetton, Dell ou Nike s'appuient sur des modes d'intégration différents de la possession du capital, mais atteignent un degré de cohérence stratégique tout à fait comparable à celui d'une entreprise capitaliste classique[8]. Elles possèdent en particulier les systèmes de gestion de la chaîne logistique les plus performants.

Elle est un ciment alternatif au ciment capitalistique pour les nouvelles entreprises en « réseau ».

Ainsi paradoxalement, on avait imaginé que l'eBusiness allait augmenter la capacité à changer de fournisseurs plus rapidement, et c'est l'inverse qui s'est produit et continuera vraisemblablement à se produire : l'investissement informatique s'effectue dans la durée, avec les partenaires privilégiés, pour qui l'on bâtit un système commun.

Certains chantiers nécessiteront au préalable une réflexion d'ordre déontologique.

Inversement, il existe des limites à l'utilité de l'informatisation.

Inversement, il est des situations où l'informatisation ne sert aucun dessein tangible. L'idéologie du « pervasive computing[9] », chère à certains, ne s'applique pas à tout. On découvre ainsi l'avance de phase des réseaux 3G, dont aucun service au consommateur ne peut encore justifier le coût d'investissement, ou les limites du « one-to-one » et de la personnalisation. Les systèmes informatiques de relation client sonnent le glas des arbitrages historiques entre richesse et impact de l'information, puisqu'ils ouvrent la possibilité de diffuser automatiquement, donc massivement, du contenu personnalisé[10]. Là où un tissu de plusieurs dizaines de milliers de conseillers financiers était la condition sine qua non d'une relation riche, il est désormais possible d'effectuer un conseil personnalisé massif sur la base d'informations de profils. Ces possibilités font néanmoins peur et soulèvent des problèmes déontologiques non encore réglés : protection de la vie privée, risques inhérents à la création de bases centralisées de profils...

Un alignement stratégique est nécessaire.

Car qui nous veut du bien au final ? Cette extension fulgurante des systèmes de scoring hérités du monde bancaire à tous les secteurs nécessite des réflexions plus poussées sur la relation qui unit une entreprise à ses clients. Celle-ci doit demeurer bilatérale, gagnant/gagnant. Les gains d'un système de Relation Client doivent donc être partagés avec celui-ci !

Des conséquences inattendues sur le marché informatique…

Ces différents écueils ont fait quelques dégâts collatéraux, dont les éditeurs de logiciels ont été les premières victimes. Progiciels de xRM (portails, relation client, relation partenaire, eCommerce), places de marché, gestion de la chaîne logistique, gestion de conception collaborative et tant d'autres, ont fait les frais de leur avance de phase sur le marché. Dans ces conditions, seuls les gros acteurs et leurs premières lignes de partenaires ou sous-traitants peuvent survivre.

Cet ensemble d'écueils a fait exploser la bulle Internet et laisse derrière lui un marché de l'édition déprimé.

Les principales vraies innovations des cinq dernières années autour des notions de relation client, chaîne logistique, achats, ou conception de produits ont été intégrées aux offres des mastodontes du secteur (Oracle, SAP, PeopleSoft…) par acquisition ou développement interne. La convergence ERP/CRM/SCM/PLM/SLM[11] annonce des suites de gestion intégrées et modulaires pour toutes les industries. Seules la banque et l'assurance restent encore

8 : Frédéric Fréry : « Benetton ou l'entreprise virtuelle ».
9 : Informatique diffuse.
10 : « Blown into Bits : How the New Economics of Information Transforms Strategy » (ou comment la nouvelle économie de l'information transforme la stratégie), de Philip Evans et Thomas Wurster.

Concentration dans les couches hautes : l'ERP à tout faire.

partiellement à l'écart de ce mouvement massif de progicialisation. Pour combien de temps ? Cette inflation des progiciels d'entreprise suit un second mouvement de fond : la convergence vers des socles d'infrastructure standard : J2EE et .Net pour l'exécution des traitements, bus EAI pour l'interopérabilité entre modules, outils PKI[12] pour la sécurisation des échanges, etc.

L'intégration des progiciels d'infrastructure est la seule réponse possible à l'ascension du logiciel libre sur ce credo.

Parallèlement, la standardisation dont se nourrit le marché depuis l'avènement de TCP/IP, annonce fatalement de grands bouleversements dans l'industrie du logiciel. Vingt ans après les premières communications entre ordinateurs via ce standard[13], nous sommes en passe de rendre aussi transparente une conversation sécurisée et sémantiquement riche[14]. Cela peut paraître long pour une industrie qui se targue des plus forts taux d'innovation, mais c'est finalement le temps nécessaire à la coordination d'un secteur. Pour autant, il est difficile de croire que les promoteurs de standards comme J2EE pouvaient imaginer à quel point une norme pouvait accélérer la concentration du marché. Il y avait une dizaine de moniteurs transactionnels avant 1998, tous plus onéreux et spécifiques. Il y a moins de cinq acteurs aujourd'hui, en comptant bien sûr le logiciel libre, fournisseur désormais incontournable de couches basses : des logiciels comme Linux, JBoss ou PostgreSQL[15] vont continuer à tailler des croupières aux éditeurs d'infrastructure classique, même après l'effondrement de leurs tarifs.

Pour contrer cette offensive, une course à l'intégration va s'engager. Car entre un serveur d'applications payant et un gratuit, le choix du gratuit risque de s'imposer dès lors que l'on aura franchi la barrière psychologique qui nous pousse encore à croire que l'un est forcément moins bien que l'autre. La contre-offensive consiste à proposer le socle de base - à la limite gratuitement -, en y associant une offre de modules et d'outils connexes.
C'est le mouvement qu'a clairement initié Microsoft avec sa plate-forme .Net, qui consacre la notion de Progiciel d'Infrastructure Intégré : transactionnel, persistance, sécurité, échanges, et outillage cohérent de conception, développement et exploitation. IBM et Oracle poursuivent également cette démarche.

Les efforts des éditeurs pour rendre plus accessibles leurs produits, qu'ils soient applicatifs ou techniques se heurtent encore au constat que seul un projet informatique sur quatre atteint ses objectifs dans le respect de son budget et de ses délais.

Restaurer la confiance nécessitera d'augmenter le taux de réussite des projets

Taux d'échec des projets informatiques en baisse, mais toujours inquiétant.

L'étude du Standish Group pour 1998 montre une amélioration par rapport à 1996 puisque le taux d'échec des projets tombe de 40 % à 28 %. En revanche, encore près de la moitié des projets sont en retard et/ou hors budget, ce qui laisse seulement un quart de projets pleinement réussis.
Ceci n'est pas une fatalité, puisque certains secteurs comme la Distribution parviennent à un taux de succès de 60 %.

L'incompréhension chronique entre fonctionnels et informaticiens à l'origine de ces statistiques déplorables.

Quelles sont les causes de telles statistiques ? En premier lieu les problèmes de pertinence des projets dans le contexte organisationnel de l'entreprise déjà évoqués : modèle économique défaillant et absence de gestion du changement.

La seconde cause majeure semble se situer dans l'incompréhension chronique entre équipes fonctionnelles et équipes informatiques. Lorsqu'un architecte en bâtiment produit le plan d'un édifice, le client final et le maître d'ouvrage savent à quoi s'attendre, tandis que le maître d'œuvre

11 : Respectivement Enterprise Resource Planning, Customer Relationship Management, Supply Chain Management, Product Lifecycle Management, Service Lifecycle Management.
12 : Public Key Infrastructure : systèmes permettant la confidentialité et l'intégrité d'échanges électroniques à l'aide de procédés cryptographiques.
13 : Voir RFC 793, septembre 1981.
14 : Notamment au travers des standards promus autour des Web Services.
15 : Respectivement système d'exploitation, serveur d'application Java et base de données relationnelle en Open Source.

sait généralement ce qu'il lui reste à faire. La raison ? Les plans. Ils instituent une représentation et un vocabulaire commun d'une part, et permettent de créer un référentiel détaillé du projet d'autre part.

Pourquoi ne pas appliquer à l'informatique les méthodes éprouvées dans d'autres secteurs ?

L'architecture crée les conditions du dialogue entre les intervenants du projet.

L'architecture, littéralement « l'Art de construire des édifices », est avant tout l'art de communiquer. Du plan commercial en 3D, au plan détaillé des évacuations d'eau jusqu'au nombre de fenêtres et de tonnes de béton à fournir, l'architecture aura servi de support aux activités des vendeurs, des maîtres d'ouvrage, et des différents maîtres d'œuvre. L'architecture est ainsi le vecteur de la **Qualité** et de la **Coordination** du projet. Imaginer construire un immeuble sans plans suffit pour avoir une idée du niveau de Qualité et de Coordination que l'on obtiendrait...

En informatique toutefois, le problème se corse légèrement. Les plans d'un Système d'Information (SI) n'ont qu'un lointain rapport avec leurs cousins des Travaux Publics. Les SI ont en effet une dimension dynamique, structurée autour des cas d'utilisation, et qui nécessite l'utilisation de différents niveaux de représentation mêlant processus, acteurs, systèmes, données, etc. Michel Pébereau, Président de BNP Paribas déclarait ainsi « L'informatique bancaire, c'est un peu les cathédrales du Moyen-Age, commencées en style roman et terminées en gothique flamboyant. Les architectes sont morts depuis longtemps et plus personne n'a les plans »[16].

Les plans de nos Système d'Information n'existent pas.

Face à cette complexité, le réflexe qui a prévalu jusqu'à présent a été de ne bâtir que des plans de très bas niveau, c'est à dire des plans d'architecture technique. Réseaux, firewall, 3-tiers, bus, hub, et maintenant peer-to-peer sont autant de représentations qui permettent de décrire la réalité physique d'un système. Mais elles achoppent dans une mission de communication englobant toutes les facettes d'un projet, notamment les plus fonctionnelles. La bonne nouvelle vient de la « montée dans les couches », tendance de fond que vit le secteur informatique depuis son apparition, et notoirement alimenté par la standardisation. Elle nous conduit inexorablement à maîtriser les systèmes à un niveau de plus en plus élevé.

Nous aborderons donc l'architecture dans sa globalité, alors qu'elle n'est réellement présente aujourd'hui qu'à un niveau technique. Du reste, si nous disposions de cartes de Systèmes d'Informations étendus, nous serions probablement surpris d'en constater la complexité, la redondance, voire l'illogisme. Conserverions-nous ces systèmes actuels de places financières, de facturation télécom ou ces réseaux d'administrations en les améliorant ou serions-nous amenés à les remplacer ? Les programmes fondamentaux du cerveau humain (son architecture donc) semblent évoluer de moins en moins avec l'âge, est-ce également le destin de nos Systèmes d'Information ? C'est-à-dire devenir des êtres complexes mus par une machinerie simpliste ? L'arbitrage entre rénovation et reconstruction est un dilemme récurrent de l'architecte.

Ainsi, en informatique comme dans le BTP, l'architecture est une méthode susceptible d'augmenter la maîtrise des chantiers d'envergure. Par sa fonction de vecteur de communication, elle concerne tous les acteurs impliqués dans les projets, au premier rang desquels figurent les donneurs d'ordre. Nous montrerons également que les investissements qu'elle nécessite se rentabilisent par l'augmentation de la qualité et la diminution des dépenses de coordination.

16 : Les Echos, 12 décembre 2001

Plus c'est compliqué à l'intérieur, plus ce sera compliqué à intégrer avec l'extérieur.

L'absence de démarche d'architecture de SI n'a pas nui à la mise en place unitaire d'applications. Progiciels clés en main et méthodologies de développement de plus en plus éprouvées ont permis d'améliorer la productivité de ces investissements. Mais cette capacité à déployer des modules de plus en plus complexes a laissé de côté la problématique de leur fonctionnement au sein d'un système d'information qui les coordonne. Par « capillarité », cette intégration entre modules se complexifie elle aussi.

L'architecture est l'art d'amener la bonne information, au bon endroit, dans la bonne forme, et de manière sécurisée.

La dématérialisation complète de chaînes de traitements nécessite une intégration des données référentielles, de la sémantique (le vocabulaire), et de la sécurité. L'architecture de SI est l'art de relier les pièces d'un puzzle, tandis que celui de créer les pièces individuelles est du ressort de l'architecture de code. Architecturer, c'est permettre d'amener la bonne information, au bon endroit, dans la bonne forme, et de manière sécurisée. Simple problème technique ? Malheureusement non.

Bâtir les futurs systèmes de remboursement de prestations médicales, d'achat de voyages à la carte, ou de règlement/livraison cross-border de produits financiers nécessitera de faire interopérer des applications de diverses générations, qui n'ont que peu de choses en commun.

Un seul challenge pour les SI de prochaine génération : augmenter leur capacité d'interopérabilité.

Par exemple, comment mettre en place un service qui préviendrait automatiquement par SMS toute personne dont l'avion subit un retard ou une annulation (embaucher des gens pour le faire ne serait probablement pas assez rentable) ? Il faut que les systèmes de réservation interopèrent automatiquement avec un réseau de téléphonie. Plus précisément, il faut que ces systèmes de réservation en back-office sachent remonter ce type d'informations au front-office, qui se chargerait de propager l'information aux clients concernés. Sachant que le distributeur du billet n'est pas forcément le producteur du trajet en avion, l'équation devient multi-acteurs, ce qui ne fait qu'ajouter à sa complexité. Ce type de problème n'a de solution qu'au travers d'une démarche de spécification étendue normalisant notamment le cycle de vie d'évènements d'entreprise comme ce banal « retard d'avion ».

La technologie n'est pas un messie, elle fait simplement partie de l'histoire …

Heureusement l'archange Technologie vient à notre secours, me direz-vous. La technologie aide, certes. EAI, infrastructures 3-tiers et outils de sécurité nous assistent dans la tâche de construction de ces « espaces de confiance », cependant ils ne suffisent pas en soi. Ce ne sont que des boîtes à outils finalement ; nous avons encore trop de latitude pour amener la bonne information au bon endroit (quelle information envoyer, à quel moment ? selon quels protocoles ? quelles méthodes de routage, statiques, dynamiques ?), dans la bonne forme (quel format ? quelle sémantique ? quelles données référentielles ?), de manière sécurisée (qui me contacte ? est-il en droit de me communiquer cette information ? est-ce bien celle qu'il a émise ?).

Interopérer impose une maîtrise de l'architecture technique ET fonctionnelle.

Attention donc aux chimères technologiques, qui alimentent illusions et malentendus. Oui, les standards émergents des Web Services vont améliorer l'interopérabilité, mais ils ne constituent qu'un élément de base de la communication, ne s'attaquant pas encore aux problèmes complexes de sémantique, qui ne trouveront pas de solution dans un carcan uniquement technique. Un produit dérivé bancaire ne se modélise pas comme un billet de train ou une commande de trois palettes de parpaings.

Pour nous assister dans la construction d'architectures applicatives, nous allons utiliser et promouvoir un certain nombre de pratiques standard, que nous nommerons patterns dans la suite. Comme la fenêtre en PVC de 2m sur 1m20 est une solution standard à l'ouverture d'une

Des « plans standards », pratiques communes de la profession, permettent de résoudre des problèmes récurrents.

pièce sur l'extérieur ; la priorité à droite, le feu de signalisation ou le rond-point, sont trois solutions possibles pour un croisement ; des patterns d'architecture, tels que la notion de royaume et d'émissaire, ou le concept de noyau sont respectivement des solutions standards à la communication inter-SI et intra-SI. Ces patterns nous permettent de décliner des architectures semblables quel que soit le secteur d'activité, diminuant de fait leur coût.

Ces similitudes nous poussent à penser que les implémentations évoluées de ces patterns constitueront le futur des Progiciels d'Infrastructure Intégrés.

La gestion du changement s'appliquera aussi aux DSI

Adapter la DSI à de nouvelles priorités.

Dans les entreprises où le niveau de synergies entre branches ou vis-à-vis de l'extérieur est élevé, il faut bien être conscient que l'interopérabilité des Systèmes d'Informations (SI), support de ces synergies ne se créera pas d'elle-même. Pour augmenter la capacité d'interopérabilité du SI, les DSI vont devoir se doter d'une organisation susceptible d'accueillir des projets mixant ressources internes et externes, tout en conservant une forte capacité de maîtrise transverse. Parce qu'elles étaient vues sous un angle purement technique, donc sans sponsor de haut niveau, les initiatives transverses actuelles, comme le déploiement de référentiels ou de bus d'échanges ont jusqu'à présent périclité. Les promouvoir au rang de priorité stratégique leur confèrera une toute autre dimension.

Voila par le menu les thèmes abordés dans cet ouvrage, nous invitons le lecteur à les parcourir à sa guise selon son profil, puisqu'il y trouvera, nous l'espérons, des réponses à des questions d'ordre organisationnel, fonctionnel et bien sûr technique.

Pierre Pezziardi

II. Démarche d'architecture d'entreprise

Le défi qui se pose à une entreprise est de savoir **quels sont les services informatiques adaptés à son contexte stratégique**. En second lieu, elle s'interrogera sur ceux devant être assurés **à l'échelle de l'entreprise (ou du groupe)**, ceux qu'il convient de laisser aux soins des **unités opérationnelles (ou des filiales)**, et ceux qu'il convient d'**externaliser**.

Mille et une DSI

Les DSI sont confrontés à trois grandes catégories de projets, dont les objectifs sont radicalement différents.

Ce contexte stratégique, varié selon les secteurs et le temps (!), induit différents types de missions pour les DSI. Ces missions peuvent se définir selon trois types d'approches : l' « utilité », la « dépendance », et le « développement ». Ces missions vont « de l'absence totale de services d'infrastructure dans la société à l'offre de services étendus, disponibles dans toute l'entreprise au sens large, y compris dans les unités opérationnelles, auprès des fournisseurs et des clients »[17].

Dans l'approche « **utilité** », les services informatiques ont pour objectif essentiel de **réduire les coûts**, par la rationalisation du patrimoine informatique ou l'informatisation de processus existants. Comptabilité groupe, mutualisation de référentiels de sécurité, ou fusion de systèmes métier sont des projets types dans cette stratégie. Une DSI opérant dans ce mode gèrera un portefeuille de projets dits « Tels Que Construits »[18], car connus au sens métier, et uniquement pilotés par le retour sur investissement précis (ROI[19]) que l'on souhaite en attendre.

Le calcul du ROI d'un projet informatique n'a de sens que dans un contexte stable et connu. Les projets innovants devront être pilotés à l'aide d'autres métriques.

Une approche « **dépendance** » répond quant à elle à une politique spécifique de l'entreprise. Ce ne sont plus simplement des économies qui sont recherchées, mais de **nouvelles opportunités à retour rapide**. Un constructeur automobile pourra ainsi connecter ses concessionnaires à ses systèmes internes, leur donnant accès aux informations produits, au suivi de clientèle, à la gestion du SAV, au commissionnement, etc. Nous parlerons de portefeuille de projets « Mieux Que Construits », puisqu'il s'agit d'améliorer des systèmes existants pour les ouvrir à des partenaires, ou les intégrer à de nouveaux processus par exemple. Dans ce cas, le pilotage par le ROI peut s'avérer plus délicat dès lors que des hypothèses doivent être introduites dans le calcul : amélioration de l'image chez les distributeurs, augmentation du coût de sortie des clients, etc.

Enfin, l'approche « **développement** » conduit à un **surinvestissement informatique** vis-à-vis des besoins immédiats. Ils concrétisent une stratégie à long terme. Une banque s'engouffrant dans la bataille de la fourniture de services bancaires constituerait ainsi une plate-forme mutualisée de distribution de ses produits, pour attaquer plus rapidement ce marché. Dans ce mode, une DSI est amenée à gérer un portefeuille de projets dits « Autres Que Construits », c'est-à-dire innovants au sens où ils informatisent une fonction qui n'existe pas encore dans l'entreprise. Le « business model » étant hautement instable, le pilotage ne peut s'effectuer que par limitation des risques, ce qui conduit souvent à une approche « bocal » (ou « nursing »), isolant le projet du reste de l'entreprise, dans un premier temps.

17 : Un système d'Information pour être compétitif, de Peter Weill et Marianne Broadbent.
18 : Catégorisation introduite par Jean-Pierre Corniou, actuel Président du CIGREF dans son ouvrage « La société de la connaissance » Ed. Hermès.
19 : Return On Investment (terme consacré).

A quoi sert une DSI ?

La DSI a pour rôle de décliner la stratégie de l'entreprise au plan informatique.

Après avoir analysé en détail plus de cinquante entreprises comprenant plusieurs unités, l'étude de Peter Weill et Marianne Braodbent conclut : « les bonnes décisions sur les systèmes d'information reposent sur une **excellente appréhension du contexte stratégique.** De plus, cette **appréhension doit être articulée, communiquée et partagée** dans l'entreprise pour mettre en évidence la relation entre la stratégie à long terme et la capacité de l'infrastructure ».

Selon nous, cet aspect communication est fondamental si l'on est sensible à l'argument que **rien ne peut finalement être imposé par la force**.

Le contexte stratégique d'un assureur pourra ainsi susciter un fort besoin dans de l'outillage de gestion de la relation client. Ne pas investir serait suicidaire dès lors que tous les concurrents améliorent sensiblement leur qualité de service perçue via ce biais. De tels systèmes vont utiliser, puis progressivement remplacer, d'autres systèmes plus anciens, que leurs auteurs vont a priori défendre. Ce renouvellement permanent doit être expliqué et appréhendé pour limiter les conflits inhérents au changement.

Bâtir la stratégie du Système d'Information et la communiquer

Communiquer cette stratégie est fondamental pour faire adhérer aux changements qu'elle nécessite.

De quoi rêverait un dirigeant pour enfin comprendre ce qui peut bien se passer derrière les portes des salles machines et des bureaux d'études informatiques ? Probablement d'un **tableau de bord structuré, contenant des indicateurs variés** sur l'adéquation des systèmes aux besoins, la qualité de service, la capacité de maintenance des systèmes, l'historique des investissements… L'objet n'est pas ici de vous proposer comment bâtir une telle « war room »[20], elle est encore bien hypothétique.

Pour autant, maintenir une architecture globale des systèmes, permet d'entamer la structure d'un tel tableau de bord. Cette première étape permettra par la suite d'y agréger toutes les informations susceptibles d'aider aux décisions d'allocations de ressources. A commencer par les informations de **coût** et de **qualité de service technique**, les plus simples.

L'architecture du SI est un calque grâce auquel il est possible de piloter collégialement les projets informatiques.

Les propositions de projets issues des unités opérationnelles seront ainsi confrontées à ces indicateurs. L'idée étant de pouvoir répondre plus facilement et de façon plus juste à des questions comme « faut-il plutôt installer un nouveau système pour mes vendeurs ou pour mes acheteurs ? » ou bien dans l'infrastructure, « un référentiel client pour mieux gérer la continuité de ma relation ? Un système d'approvisionnement automatisé et centralisé ? Une plate-forme d'acquisition de flux (EDI, Swift...) mutualisée ? » Le tableau de bord, structuré en domaines et sous-domaines de l'entreprise, permet de positionner chaque nouvel investissement vis-à-vis des indicateurs de son domaine. On privilégiera, sous réserve des enjeux stratégiques métier, les cases « rouges », qui peuvent l'être pour diverses raisons : pauvreté fonctionnelle (dans l'absolue ou face à la concurrence), faible qualité de service, vétusté des systèmes, etc.

De l'autre côté de la hiérarchie, l'architecture devient un **outil de communication** vers les employés, et les équipes informatiques en particulier. Comme on attend mieux sur le périphérique parisien dès qu'un panneau vous informe qu'il vous reste vingt minutes à poireauter, on se trouve également mieux lorsque l'on se situe par rapport à l'univers informatique de la société. On finit par y adhérer quelque part.

20 : Les organisations du XXIe siècle, de Marc Lemarignier et Jessica Scale.

Créer et promouvoir les ouvrages transverses du SI

Les ouvrages transverses du SI peuvent avoir une utilisation directe, comme une comptabilité groupe, ou indirecte, comme un référentiel métier

Les services informatiques transversaux se sont longtemps limités aux centres d'exploitation et aux services techniques (poste de travail, réseau, firewall…). L'évolution des technologies et de la standardisation permet de proposer des services applicatifs transverses. A commencer par les ERP : ressources humaines, achats, comptabilité. Ces services ont une utilisation directe, ce qui n'est pas le cas de certaines fonctions applicatives « d'infrastructure », qui bien que fondamentales, n'ont qu'une utilisation indirecte : par exemple, les référentiels (utilisateurs, tiers, et produits essentiellement), ou les différents niveaux possibles de plate-forme employés - plate-forme clients, plate-forme fournisseurs, ou plate-forme partenaires que nous détaillerons plus loin.

Pas d'ouvrage transverse sans proposition de valeur perceptible pour les projets.

Ces ouvrages sont clairement de la responsabilité des DSI, qui doivent également en assurer la promotion et le support auprès des projets. Offrir un service attrayant est le seul moyen d'éviter l'échec de ce type d'initiatives, souvent jugées lourdes et complexes par les projets, qui préfèrent travailler indépendamment, loin des « usines à gaz ». Les chantiers transverses doivent proposer une valeur perceptible pour les projets et être accessibles simplement.

Insuffler et contrôler les orientations des projets

Enfin, loin de vouloir contrôler tout ce qui est réalisé dans le SI, ce qui est totalement vain, il faut néanmoins se donner les moyens d'insuffler et de contrôler les principaux partis pris architecturaux des projets. Qu'ils soient de dimension fonctionnelle (quel découpage de domaines ? quelle utilisation des ouvrages transverses ? quelle méthodologie de conception ?) ou bien de dimension technique (méthodologies d'analyse, de développement, de test et d'intégration, architecture, standards technologiques…).

Gérer l'interaction de l'entreprise avec les organismes de standardisation du secteur

Les normes d'interopérabilité sectorielles jouent un rôle de plus en plus important dans les entreprises, en imposant des schémas fonctionnels et techniques aux activités les plus tournées vers l'extérieur.

Une conséquence de la globalisation est l'augmentation importante du nombre de transactions dénouées électroniquement. Tous les secteurs ont aujourd'hui besoin d'organismes de standardisation qui définissent les normes d'interopérabilité pour ces transactions électroniques. Les réalisations de Swift dans la finance, de l'ITU dans le monde des télécommunications ou de l'UN/CEFACT dans l'industrie, sont des exemples significatifs de l'importance de ces consortiums. Pour une grande entreprise, il est fondamental d'y être représentée, d'une part pour recueillir les meilleures informations sur les normes actuelles et à venir, mais aussi pour influer sur certains choix dans ces normes.
En disposant d'une compétence importante sur ces aspects, la DSI renforce également sa légitimité auprès des unités opérationnelles.

La DSI, une SSII comme les autres ?

Pour justifier sa primauté sur les projets informatiques, la DSI doit différencier ses compétences vis-à-vis des offres de services extérieures…

Qu'est-ce qui différencie un chef de projet, un concepteur, un développeur, un support exploitation issu de la DSI de son homologue de SSII ? La valeur créée par l'internalisation de telles ressources n'est pas toujours évidente à déterminer.

Pourtant, les missions de la DSI que nous évoquions, à savoir bâtir la stratégie du SI et la communiquer, créer et promouvoir les ouvrages transverses du SI, insuffler et contrôler les orientations des projets, ou gérer l'interaction de l'entreprises avec les organismes de standardisation du secteur, sont des missions de longue haleine, à inscrire dans une continuité.

Deuxième point essentiel, la nécessaire ouverture des DSI vers l'extérieur, au-delà de la trop traditionnelle dualité Maître d'œuvre/Maître d'Ouvrage héritée du bâtiment. En particulier, comment vont s'organiser les missions d'urbanisme ou de construction de systèmes transverses impactant de nombreux processus existants ? Comme l'évoque François Tabourot, l'apport des Directions Métier est alors fondamental. Une démarche d'urbanisme centrée sur la l'optimisation du rapport coût/qualité du SI peut à la limite être menée par la DSI seule.

Maîtriser une méthodologie de construction des projets informatique est un pré-requis à cette compétitivité.

Une démarche d'urbanisme dans la durée doit modifier plus profondément les manières de faire existantes dans l'entreprise, pour que tout nouveau projet s'intègre dans les « plans ». Ceci ne peut être le fait unique d'une cellule imposant des contraintes aux projets (en avaient-ils vraiment besoin ?), mais bien la **maîtrise diffuse d'une méthodologie**, c'est-à-dire une manière commune d'aborder les projets : conception par les processus, modélisation, architecture d'intégration dans le SI, etc. En somme, on urbanise plus par le bas que par le haut, ce qui signifie finalement que **l'urbanisme c'est de la formation avant tout**.

Un bureau d'architecture, où siègent fonctionnels et techniques, peut assumer une mission de gardien du temple.

Au chapitre du « comment », la solution unique n'existe pas. Dans Urbanisation du business et des SI, Gérard Jean introduit par exemple une « maîtrise d'ouvrage stratégique », émanation très « jacobine » et « métier », destinée à dessiner les plans du SI à long terme. Pourquoi pas ? Ou pourquoi pas une DSI Groupe réunissant un « Comité SI » avec les directions métier ? Le seul point important est de parvenir à **élargir la réflexion Système d'information au-delà du cercle des experts techniques**.

Compte tenu du caractère opérationnel de certaines missions, dont la mise au point et la diffusion d'une méthodologie projet de bout en bout, un **bureau d'architecture permanent** peut être une réponse à ces sujets.

Une activité de haute précision

Tout l'art de la méthodologie est de se positionner en facilitateur plus qu'en « commissariat au plan ».

Il faut être très vigilant aux limites de la centralisation et à la nécessaire subsidiarité des équipes projet. **Imposer des structures trop contraignantes crée les conditions mêmes de la subversion**. Ainsi, maintenir une architecture globale ne signifie pas imposer aux projets des lourdeurs administratives supplémentaires, en leur infligeant divers dossiers d'architecture, mais au contraire en leur apportant un soutien. C'est la vieille histoire du client/serveur, où un prétexte technologique a permis de décentraliser les projets, trop longtemps soumis au joug de l'informatique centrale ! Mais ce qui a été gagné en rapidité a été immédiatement, et pour longtemps, perdu au niveau de l'interopérabilité, donc au détriment de l'agilité du SI.

De plus, dans certains grands groupes, plusieurs structures de DSI adossées aux différents métiers cohabitent. Dans les entreprises où le besoin de synergies entre ces métiers augmente, des DSI Groupe ont émergé. Le partage des prérogatives entre ces acteurs n'est généralement pas simple. Car chacun défendant son niveau de transversalité, il arrive que les champs interfèrent. Entre annuaire groupe et annuaire branche, entre eProcurement groupe et solution d'achats locale, il est souvent difficile de trancher entre le plus global, mais aussi le plus long et le plus cher, et le moins global, mais aussi le plus rapide et rentable à court terme. DSI et DSIG sont probablement condamnées à vivre en compétition, selon les axes de développement plus ou moins long terme promus par la Direction.

Conclusion

Pour assurer ces différentes missions, les entreprises auront besoin de mettre en place un **Bureau d'Architecture du Système d'Information**, cellule dédiée rassemblant des **compétences fonctionnelles et techniques**, et dont le rôle sera :

- de dresser les plans globaux du SI et de ses frontières immédiates, et les communiquer,
- de faire appliquer des normes d'urbanisme et d'interopérabilité au niveau des projets, en promouvant une méthodologie projet de bout en bout,
- de promouvoir opérationnellement les ouvrages transversaux d'infrastructure,
- de recueillir les standards d'interopérabilité du secteur dans les consortiums ad-hoc, et promouvoir leurs extensions souhaitées dans le cadre de l'activité de l'entreprise.

Le Bureau d'Architecture ne doit pas se transformer en commissariat au plan, Kremlinisé dans une tour d'ivoire bien à l'abri des projets. Il doit au contraire accueillir, valoriser et diffuser les compétences issues des lignes métier, d'où provient l'essentiel de l'innovation. C'est une **communauté, un ordre, plus qu'une structure à part entière**.

III. Etat de l'art des principes d'architecture

Architecture – « Art de concevoir et de construire un bâtiment selon des partis esthétiques et des règles techniques déterminés » (éditions Larousse)

Dans le domaine de l'Informatique comme dans celui du Bâtiment, l'architecture a ses écoles, ses styles, ses courants… Mais l'informatique est une science encore jeune, elle a ainsi vu se succéder en quelques décennies plusieurs méthodologies d'architecture, qu'il s'agisse de l'architecture logicielle ou de l'architecture de SI…
Si dans le domaine de l'architecture logicielle un consensus s'est créé ces dernières années autour du paradigme objet et des méthodologies basées sur UML[21] (Unified Process ou eXtreme Programming notamment), en revanche dans le domaine de l'architecture de SI aucune méthodologie n'a réussi à s'affirmer avec succès : force est de constater que ces méthodologies se sont le plus souvent limitées à des projets spécifiques sans parvenir à se généraliser à l'échelle du SI.
C'est pour mieux estimer la difficulté à définir et pérenniser des principes d'architecture de niveau SI que nous proposons ici un état de l'art critique des principales méthodologies d'architecture, de l'architecture logicielle à l'architecture de SI. L'analyse des réussites et des échecs nous permettra d'une part de modérer nos prétentions - il n'existe très certainement pas de méthodologie unique d'architecture, la nôtre ne faisant pas exception -, et d'autre part, de justifier les principes fondateurs de la démarche d'architecture de SI que nous proposons dans cette ouvrage.

L'Architecture Logicielle

L'architecture logicielle se consacre à architecturer et concevoir le « code » d'une application à partir de ses spécifications fonctionnelles. L'architecture de SI se consacre à architecturer et intégrer un ensemble d'applications et de référentiels à partir de spécifications fonctionnelles définies au niveau plus global du SI.

L'architecture logicielle est un domaine dans lequel s'est épanoui au cours des dix dernières années le paradigme objet ainsi que les langages et méthodologies associés (UML, xUP, Design-Patterns, XP…). L'état de l'art en la matière est maintenant relativement stable, et les méthodologies d'ingénierie logicielle dignes des méthodologies de production industrielle…

Mais pourquoi consacrer un chapitre à l'architecture logicielle dans un livre blanc consacré à l'architecture de SI ? Pourquoi s'intéresser à un domaine qui s'adresse a priori plutôt aux équipes de développement et pour lequel il existe à ce jour une littérature abondante ?

C'est précisément parce qu'il s'agit d'un domaine d'architecture dans lequel une méthodologie s'est affirmée avec succès que nous estimons utile de nous y attarder quelques instants, sans pour autant vouloir faire un énième ouvrage sur UML/UP… Nous le verrons dans le chapitre suivant, les méthodologies d'architecture de SI pêchent par manque de pragmatisme, par la croyance illusoire aux démarches « top-down », et il nous apparaît important de souligner en quoi les méthodologies d'architecture logicielle basées sur le paradigme objet se démarquent de ces approches.

21 : Unified Modeling Langage.

UML

Positionnement

L'abstraction est la capacité à ne voir que l'essentiel d'un problème ; on parlera de niveau d'abstraction suivant le niveau de maille de l'analyse du problème. L'encapsulation consiste à ne s'intéresser qu'à l'interface du système et à masquer les mécanismes internes.

Les langages objet et les méthodologies orientées objet (OO) sont aujourd'hui un standard incontournable dans le domaine de l'ingénierie logicielle, les deux piliers du paradigme objet que sont **l'abstraction** et **l'encapsulation** étant particulièrement adaptés à la démarche d'analyse et de conception de systèmes logiciels.

Les premiers langages objet sont apparus au milieu des années 80 (SmallTalk, Eiffel) et les premières méthodes d'analyse OO sont quant à elles apparues au début des années 90, époque à laquelle on dénombrait plusieurs dizaines de méthodologies OO. La concentration s'est accélérée vers 1995, par la stabilisation autour de trois méthodes (OOM, OMT, OOSE) puis par la fusion en 1997 de ces méthodes dans un langage unique, UML, sous la coupe de l'OMG[22].
En terme d'analyse et de modélisation objet, UML est aujourd'hui un standard incontournable, stabilisé, industriel (pris en charge par la plupart des outils de modélisation et de développement). Au delà des maîtrises d'oeuvre, UML est également de plus en plus utilisé par les maîtrises d'ouvrage pour spécifier fonctionnellement les cas d'utilisation d'une application, pour modéliser les processus métier au niveau SI[23]…

A travers les différents cas d'usage possibles, UML reste un **langage formel de modélisation** (basé sur un méta-modèle, un formalisme rigoureux), mais n'est ni une méthodologie (on peut très bien adopter par exemple une méthodologie OMT en utilisant le formalisme UML), ni un processus de développement qui définit les étapes du processus, les acteurs, la démarche (UP, RUP[24]). En revanche, UML cadre l'analyse en proposant une démarche de modélisation basée sur **l'élaboration itérative de modèles**.

La démarche de modélisation UML

Le principe d'UML, et ce qui en fait sa force, est de représenter un système par un ensemble limité de **modèles** et de cadrer l'analyse du système en proposant :

- des **niveaux d'abstraction différents** suivant les modèles, pour maîtriser la complexité du système : modèles de use-case, modèle d'analyse, modèle de conception…
- des **différentes vues complémentaires** d'un système (diagrammes statiques, dynamiques), pour guider l'utilisation de concepts objet.

Nous ne détaillons pas ici le détail de la démarche de modélisation, le formalisme UML, ou les modèles, que le lecteur pourra trouver dans les nombreux ouvrages existants sur le sujet… Cependant dans le cadre d'une comparaison au domaine de l'architecture de SI, il est important de retenir que cette démarche repose sur trois piliers :

22 : Open Management Group (www.omg.org). L'OMG a notamment normalisé CORBA et MDA.
23 : Cf. § « Architecture de SI » plus loin.
24 : Rational Unified Process (Rational Software).

1. La démarche est **itérative et incrémentale :** le système est construit et validé par étapes. Les modèles sont élaborés de plus en plus finement, par itérations successives et allers-retours, en s'appuyant sur des niveaux d'abstraction différents et de plus en plus fins :

   - La recueil et la formalisation des besoins : diagramme de use-cases, diagrammes d'activités, spécification détaillée des différents scénarios de chaque use-case sous forme de diagramme de séquence…

   - L'analyse du domaine : identification des objets métiers, associations entre objets… On établit ici le « MCD[25] » objet de l'application.

   - L'analyse logicielle : modélisation des principaux mécanismes logiciels, identification des classes d'interface/contrôle/entités, diagrammes de séquence et de collaboration entre ces éléments pour chaque scénario d'un cas d'utilisation…

   - La conception : détail des rouages d'implémentation logiciels, en s'appuyant par exemple sur un certain nombre de patterns[26] (Design-Patterns [12], Architectural Patterns [13]...), qui proposent des « manières de faire clé en main » : Pattern « Façade » lorsque l'application doit utiliser un sous-ensemble restreint de services d'un système complexe, Pattern « Broker » pour déléguer l'invocation de plusieurs services à un objet dédié…

2. La démarche est **pilotée par les Use-Cases** : ce sont eux qui guident l'élaboration des modèles ; la recueil des besoins principaux du système est le point de départ de la démarche, et les modèles évoluent conjointement avec l'enrichissement du modèle des cas d'utilisation.

Les trois piliers fondateurs de toute démarche de modélisation basée sur UML sont l'itération en zooms successifs (abstraction), le pilotage par les use-cases, et une réflexion architecturale très en amont.

Au delà de l'analyse et de la conception, c'est l'ensemble du processus de développement qui est piloté par les use-cases, comme l'illustre le schéma suivant :

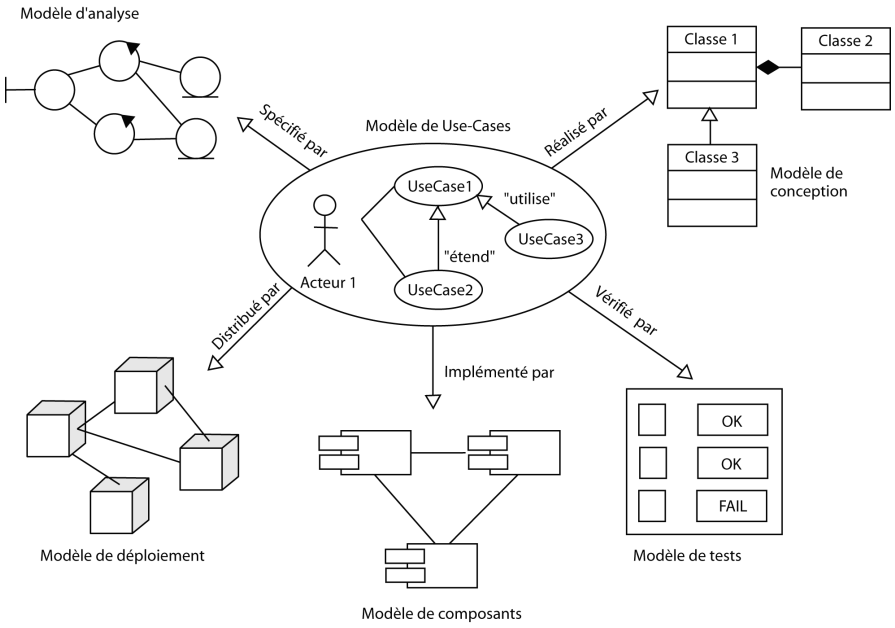


Figure 1 : UML, un ensemble de modèles pilotés par les cas d'utilisation

25 : Modèle Conceptuel de Données.
26 : www.patternsdepot.com,www.cs.wustl.edu/~schmidt/patterns.html,www.posa.uci.edu, hillside.net/patterns...

3. La démarche est **centrée sur l'architecture** : l'architecture est la clé de voûte de la démarche, elle est garante de la bonne implémentation des Use-Case, des objectifs du projet (qualité, coûts, interopérabilité, réutilisabilité), elle permet d'identifier les risques, et de justifier les choix.

Selon P. Kruchten[27], l'architecture logicielle se représente selon « 4+1 » vues :

4 vues pour représenter l'architecture logicielle selon l'angle de vue des différents acteurs. + 1 vue use case qui les pilote.

- **Vue Logique** : elle se concentre sur la modélisation des principaux éléments d'architecture et mécanismes logiciels, la définition des éléments du domaine (objets métier, diagrammes d'états de ces objets…). Elle comprend en particulier les modèles d'analyse et de conception du système, c'est la vue des équipes de conception et développement.

- **Vue Composants** (ou vue Implémentation) : elle permet d'identifier les modules (composants logiciels) qui implémentent les éléments définis dans la vue logique, de les regrouper en composants logiciels, d'identifier les dépendances d'intégration entre ces composants ; c'est la vue des équipes d'intégration.

- **Vue Processus** : en environnement multi-tâches, cette vue de l'architecture permet de définir les processus, la coordination et la synchronisation des processus, les threads d'exécution… Cette vue est optionnelle et n'est utile que dans le cas d'architectures complexes multi-tâches.

- **Vue Déploiement** : elle précise l'architecture de production (ressources matérielles, implantation des composants, pilotage…) Cette vue permettra par exemple de s'assurer que l'application répond aux contraintes de déploiement, aux exigences de qualité de service (montée en charge, temps de réponse, haute-disponibilité…) et s'intègre aux infrastructures de supervision, etc.

- **Vue Use-Cases** (vue « +1 ») : elle se concentre sur un sous-ensemble des cas d'utilisation qui ont une influence significative sur l'architecture du système. Ces use-cases structurants permettent d'identifier les fonctionnalités et contraintes importantes, les risques majeurs de l'architecture, ce sont eux qui guident l'élaboration des quatre autres vues de l'architecture, de la conception à la mise en production de l'application.

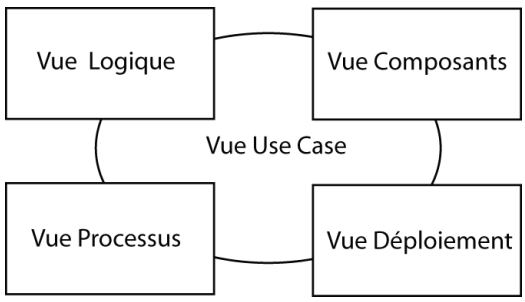


Figure 2: Vues de l'architecture logicielle selon P. Kruchten

27 : « The 4+1 view model of software architecture », IEEE 1995 – Ce modèle a servi de référence pour l'ensemble des méthodologies et processus basés sur UML.

UP, un processus générique de développement

A ce stade, nous ne parlons pas encore de processus d'ingénierie logicielle : comment mettre en œuvre cette démarche de modélisation au sein d'un projet ? Quels sont les acteurs, les activités, les documents échangés, les phases et leurs livrables ? C'est précisément le but des méthodologies de développement dérivées de UP de proposer une démarche projet de bout en bout, de la phase de recueil des besoins à la mise en production de l'application.

Sans vouloir détailler UP ou l'un de ses processus dérivés comme RUP, retenons qu'il s'agit d'un processus se basant sur les trois piliers précités (démarche itérative, pilotée par les use-cases, et centrée sur l'architecture) et proposant un découpage du projet en quatre phases :

- La phase « **Création** », dont le but est de présenter un certain nombre d'éléments qui permettront de décider ou non de l'opportunité du projet : expression des besoins (principaux use-cases), ébauche grossière et provisoire de l'architecture, organisation nécessaire, estimation des coûts de développement et du ROI, recensement des principaux risques (organisationnels, techniques) ;

- La phase « **Elaboration** » qui va préciser la plupart des Use-Cases, la définition de l'architecture de référence du système, un plan projet détaillé pour la finalisation du projet. En sortie de cette phase, on aura réalisé un prototype qui aura permis de valider les grands choix architecturaux et de traiter les risques majeurs ;

- La phase « **Construction** » dans laquelle l'application va être développée, l'architecture stabilisée, l'ensemble des use-cases définis, implémentés et testés. En sortie de cette phase, on se retrouvera avec une version « beta » de l'application.

- La phase « **Transition** » qui consiste à effectuer un beta-testing de l'application sur un nombre réduit d'utilisateurs, de corriger les bugs constatés, puis de déployer entièrement l'application (mise en production, mise sur le marché…).

UP, un processus de développement unifié qui reprend les trois piliers de la démarche UML et décompose le projet en quatre phases ; notre méthodologie d'architecture de SI reprendra beaucoup des concepts UP, notamment l'itération.

UP définit également un certain nombre d'activités (expression de besoins, analyse, conception, implémentation, tests/déploiement, gestion) que l'on va retrouver au sein des différentes phases ; les méthodologies dérivées d'UP (RUP par exemple) s'attachent à décrire dans le détail ces activités, les acteurs, les documents, le workflow entre acteurs, les livrables en fin d'activité et en fin de phase…

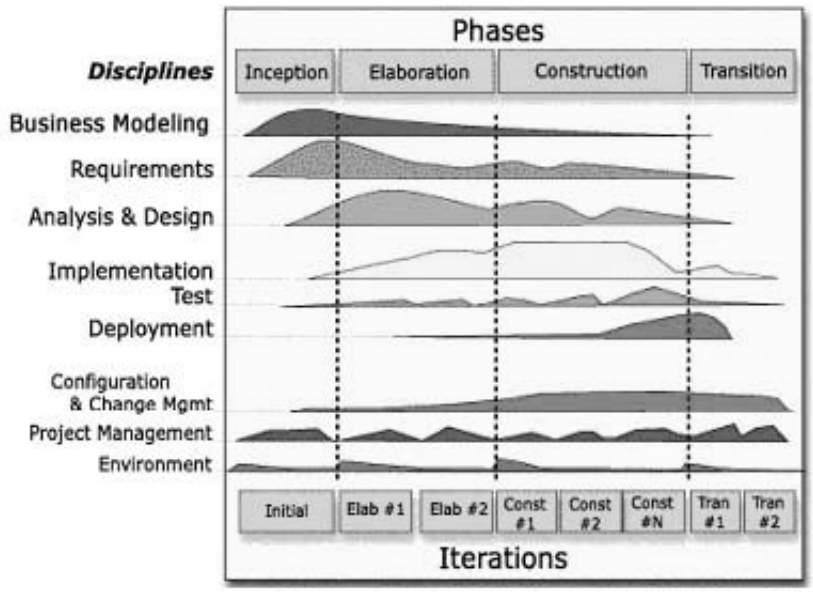


Figure 3: Exemple de Processus Unifié : RUP (source : Rational Software)

Conclusion

En ce qui concerne les activités de développement et d'ingénierie logicielle, un large consensus s'est opéré autour du paradigme objet, principalement pour les raisons suivantes :

- Utilisation d'un langage de modélisation formel et standardisé, UML ;
- Puissance et adéquation du paradigme objet (abstraction, encapsulation) pour les activités d'analyse et de conception qui permet la modélisation à des niveaux successifs d'abstraction ;
- Démarche itérative, et non séquentielle, entre les phases de recueil des besoins, analyse, conception, grâce notamment aux niveaux d'abstraction proposés par les modèles ;
- Unification du langage de modélisation UML et des langages de développement (Java, C#, etc.) autour d'un même paradigme (l'objet), ce qui favorise la continuité entre les phases de conception et les phases d'implémentation ;
- Large utilisation de patterns dans les phases d'analyse et de conception (Design Patterns, Analysis Patterns…).

En revanche, l'intégration d'UML dans les projets de développement passe par une phase non négligeable de formation voire d'évolution culturelle (passer par exemple du monde Cobol à Java ou de la spécification par l'écran à la spécification par le processus, ne se font pas sans mal), et d'autre part la mise en place de processus de type UP peut représenter un risque fort en l'absence d'un socle méthodologique au niveau entreprise. Prudence donc avant de foncer tête baissée dans l'utilisation de RUP au sein d'un projet de développement !

L'Architecture de SI

L'architecture de SI se différencie de l'architecture logicielle par les concepts manipulés : modules logiciels, classes, composants pour l'architecture logicielle ; modules applicatifs, référentiels, flux pour l'architecture de SI. En outre, l'architecte logiciel doit concevoir une architecture implémentant des cas d'utilisations centrés sur une application, alors que l'architecte de SI devra concevoir une architecture implémentant des cas d'utilisation transverses à différentes applications… Si la démarche est globalement la même (définir la structure d'un système, ses composants, sa dynamique…), architecture logicielle et architecture de SI opèrent cependant à des niveaux différents de granularité et d'abstraction.

Si l'élément de référence en architecture logicielle est la classe logicielle, elle devrait être la brique applicative en architecture de SI… Or les différentes méthodologies d'architecture de SI existantes s'attachent à structurer un SI en classes logicielles !

Et pourtant, de la même façon qu'une méthodologie d'architecture logicielle permet d'aller jusqu'aux spécifications des classes logicielles d'une application en partant des spécifications fonctionnelles, les différentes méthodologies d'architecture de SI se sont attachées à atteindre les mêmes objectifs : permettre, à partir des processus métier de l'entreprise, de décrire les spécifications techniques détaillées du l'ensemble du SI, jusqu'aux interfaces des composants et aux modèles de données…

Ces méthodologies « top-down » (du processus au code), se regroupent en deux courants principaux, l'approche « Données/Traitements » (Zachman, Merise…) et l'approche « Composants » (RM-ODP, Catalysis…) qui adresse plus spécifiquement les architectures des systèmes distribués.

Mais indépendamment de la méthodologie choisie, une démarche d'architecture commence par l'urbanisation des fonctions métier (structuration en blocs fonctionnels, échanges entre ces blocs) à partir des processus métier ; c'est cette première phase d'architecture au niveau métier de l'entreprise (processus, modèle d'information…) que nous nous proposons d'illustrer avant de nous concentrer plus précisément sur les méthodologies d'architecture de SI.

L'urbanisation des fonctions du SI

Si la définition des processus métier de l'entreprise et du modèle d'information ne relève pas explicitement des activités d'architecture mais plutôt de celles des fonctionnels, l'architecte devra en revanche, au niveau métier, urbaniser les fonctions et processus, structurer le SI en blocs fonctionnels, garantir son évolutivité fonctionnelle… Une tâche maîtrisable au niveau d'un périmètre fonctionnellement limité (projet, système autonome...) mais qui devient rapidement complexe à l'échelle du SI d'entreprise, tant le caractère transverse et étendu des processus métier est difficile à structurer.

Pour maîtriser cette complexité, l'entreprise peut se baser d'une part sur une méthodologie « standard » de **modélisation des processus** (outils, méthodes, référentiels...) partagée par les différents projets, et d'autre part, sur **l'utilisation de patterns métier**, apportant des bases de solutions à des problèmes connus et référencés.

Méthodologies de modélisation des processus

Les méthodologies de modélisation des processus métier sont le plus souvent basées sur des outils BPM ou BPR[28] du marché (Mega, Casewise…), voire sur des outils de type Visio ou Word… Il n'existe pas de standard en matière de méthodologie, chaque société de conseil ou chaque éditeur spécialisé dans le domaine propose généralement sa propre méthodologie de modélisation, qu'elle soit basée sur un formalisme UML ou un formalisme propriétaire…

28 : Resp. Business Process Modeling, Business Process Re-engineering.

Qu'il s'agisse de méthodologies de modélisation de processus ou de patterns métier, les initiatives qui réussissent sont celles poussées par les organismes sectoriels… Un exemple parmi tant d'autres, la course de ebXML le généraliste après RosettaNet le pragmatique…

Notons en revanche l'initiative UMM[29] en cours de gestation au sein de l'UN/CEFACT, dont l'objectif est de fournir une méthodologie standard basée sur UML pour la modélisation des processus métier entre une entreprise et ses partenaires ; cette méthodologie résulte de la fusion de méthodologies issues d'organisations verticales comme ITU, TMForum, RosettaNet, et a été adoptée par différents organismes horizontaux comme GCI (Global Commerce Initiative), X12 pour l'EDI, ebXML, et par certains organismes verticaux comme SWIFT pour la finance. Cette méthodologie n'est pas encore stabilisée, mais espérons que le fait qu'elle soit guidée et adoptée par des initiatives verticales soit un gage de succès et de pérennité.

Patterns métier

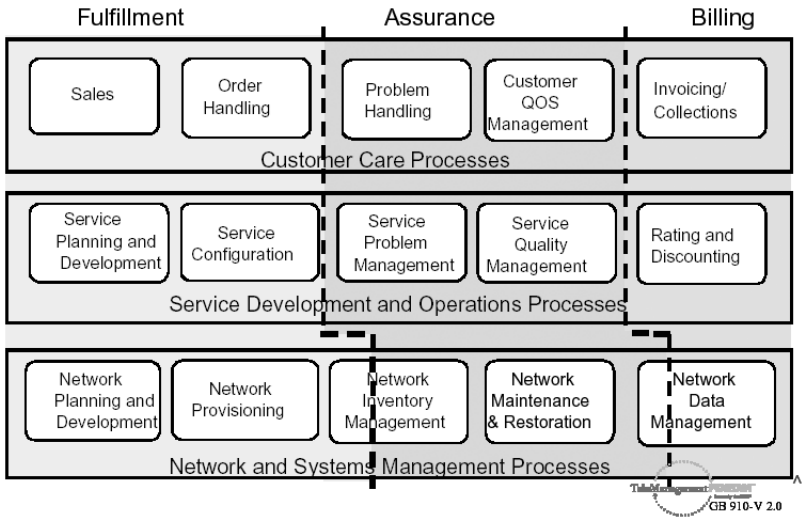
Les patterns métier utilisés pour la structuration des processus, l'urbanisation des fonctions, ou la définition d'un modèle d'information, sont généralement issus d'initiatives verticales… La raison en est simple : qui mieux qu'un consortium d'opérateurs télécoms peut définir les règles d'urbanisation du métier d'un opérateur ? Certainement pas des initiatives horizontales comme ebXML, OAG...

Plusieurs initiatives verticales ont ainsi proposé avec succès des patterns, frameworks ou modèles d'information. Citons parmi les plus connus :

- TOM ou NGOSS dans le domaine des télécommunications (voir encadré ci-après) ;
- RosettaNet dans le domaine du Supply Chain Management (SCM) ; RosettaNet est un consortium de 350 compagnies, principalement des équipementiers de matériel informatique et électronique, qui a réussi à faire adopter par l'ensemble de l'industrie un ensemble de processus collaboratifs, un dictionnaire de données et de messages, (interopérabilité sémantique), une infrastructure de communication (interopérabilité technique)…

TOM (Telecom Operations Map) est un framework pour la définition des processus métier opérationnels d'un opérateur. Proposé par le Telemanagement Forum dans le cadre du modèle NGOSS (Next Generation Operation Systems Support), TOM permet de structurer les processus opérationnels centrés sur le client de l'opérateur en trois catégories verticales : « Fulfillment » (capacité à abonner le client au service), « Assurance » (capacité à fournir le service au client), « Billing » (facturation des services au client). Pour chaque processus de ces catégories verticales, TOM propose un modèle d'implémentation « end-to-end » pour les fonctions horizontales de Customer Care, Service Development, Network Management, en détaillant les services offerts par chaque module, les enchaînements et interactions entre ces services…
La figure suivante montre d'une part les principaux modules fonctionnels du modèle TOM, et illustre comment le modèle peut aider à la structuration d'un processus « end-to-end » de facturation impliquant l'opérateur et des « Service Providers » externes.

29 : Unified Modeling Methodology.



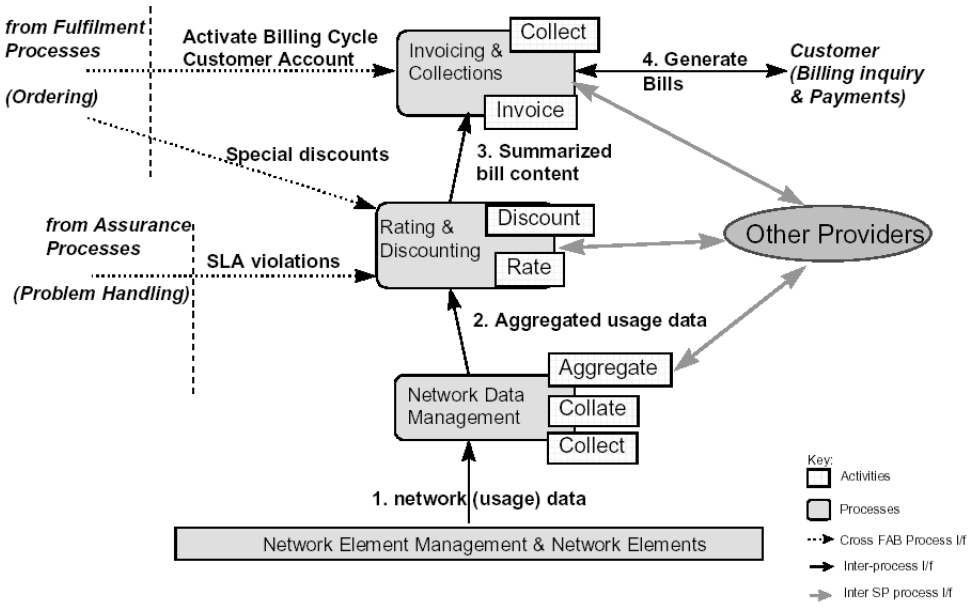


Figure 4: Le modèle TOM et un exemple d'urbanisation d'un processus de facturation

TOM évolue actuellement vers eTOM, extension de TOM à l'ensemble des processus d'un Service Provider (ASP, fournisseur de contenu 3G…) – eTOM ne se limite pas aux processus opérationnels d'un opérateur mais fournit un cadre pour l'ensemble des processus d'une entreprise (gestion de produit, stratégie, RH...), impliquant l'ensemble des acteurs externes à l'entreprise : clients, partenaires, fournisseurs... Un modèle en cours de maturation qui, avant d'être applicable à l'ensemble des entreprises, devra être éprouvé par les futurs acteurs du 3G… Patience donc !

Pour l'architecte, structurer son architecture métier en s'appuyant sur ce type de patterns va lui permettre :

- De disposer d'une vision transverse de l'orchestration des processus de l'entreprise, lui permettant de proposer des schémas d'urbanisme pertinents,
- De favoriser l'interopérabilité entre l'entreprise et ses partenaires (processus, sémantique des messages...),
- De définir les formats pivots entre blocs fonctionnels du SI (par exemple, utiliser xCBL comme format d'échange entre les applicatifs de gestion des stocks…),
- De favoriser la « progicialisation » de certains applicatifs, par intégration de progiciels spécialisés de façon standardisée sur certaines fonctionnalités (ex. : Operation Support Systems pour les opérateurs).

L'entreprise aura donc tout intérêt à utiliser au maximum ce genre de patterns lorsqu'ils existent, voire à s'impliquer dans des initiatives sectorielles de standardisation.

Architecturer le SI : l'approche « Données/Traitements »

Les premières méthodes d'analyse de SI sont apparues au début des années 70, avec la montée en puissance des applications de gestion sur systèmes centraux ; on parlait alors des méthodes analytiques (Corig, Warnier) qui consistaient à décomposer le SI de façon atomique avec un très faible niveau d'interaction entre les fonctions du système : indépendance des traitements, des données…

Au début des années 80, ces méthodes ont évolué vers l'approche systémique, qui considère le SI comme un tout cohérent avec un niveau plus élevé d'interactions entre les fonctions du système. Le SI y est vu comme un système dynamique et interactif, constitué d'un ensemble de traitements opérant sur des données en entrée et restituant des données en sortie (traitement d'un bon de commande, édition de facture…) : c'est l'apparition des méthodes d'analyse de SI orientées « Données/Traitements », comme Zachman ou Merise en France…

L'approche « Données/Traitements » centre l'analyse d'un problème sur la donnée manipulée.

Cette approche structure l'analyse fonctionnelle d'un système en séparant distinctement le « quoi » (quelles sont les données manipulées ?) du « comment » (quels traitements sont réalisés sur ces données ?). Tout au long de la phase d'analyse, le problème sera décomposé séquentiellement en fonctions, sous-fonctions, jusqu'aux traitements unitaires, en gardant continuellement cette dualité « Données/Traitements », et en modélisant exhaustivement les données et les traitements.

Le premier framework d'architecture de SI d'entreprise basé sur l'approche « Données/Traitements » fut proposé par Zachman[30] au début des années 80 : il s'agit d'un cadre unifié d'architecture qui s'étend de la modélisation métier à l'architecture de production en passant par le développement logiciel. Ce framework est encore utilisé de nos jours dans les pays anglo-saxons et intégré à de nombreux outils CASE[31] (Casewise, Popkin Software, Visible Systems...).

30 : www.zifa.com.
31 : Computer-Aided Sofware Engineering.

Le Framework Zachman se représente de façon matricielle en six colonnes et cinq lignes :

- Les colonnes adressent les aspects de l'entreprise : Quoi (que traite-t-on), Comment (comment traite-t-on), Où (aspects géographiques), Qui, Quand, Comment (aspects organisationnels),

- Les lignes adressent les vues selon les différents acteurs de l'entreprise : les enjeux stratégiques, puis le modèle métier, le modèle fonctionnel SI, les modèles conceptuels et physiques d'architecture…

- A chaque cellule de la matrice sont associés des documents formalisés d'architecture; par exemple, le formalisme Entités/Associations sera utilisé pour « Modèle Conceptuel de Données» de la cellule « Données/Conception ».

Le framework Zachman : un cadre d'architecture centré sur la donnée et associé à une démarche séquentielle qui part des processus métier pour arriver jusqu'à l'implémentation physique des systèmes…

La figure suivante montre une représentation partielle de ce framework (les colonnes Qui, Quand, et les aspects organisationnels ne sont pas représentées) :

	DATA *What*	FUNCTION *How*	NETWORK *Where*
OBJECTIVES/ SCOPE (CONTEXTUAL) *Planner*	List of Things Important to the business ENTITY = Class of Business Thing	List of Processes the Business Performs Process = Class of Business Process	List of Locations in which the Business Opeerates Node = Major Business Location
ENTERPRISE MODEL (CONCEPTUAL) *Owner*	e.g. Semantic Model Ent = Business Entity Reln = Business Relationship	e.g. Business Process Model Proc = Bus Process I/O = Bus Resources	e.g. Business Logistics System Node = Business Location Link = Business Linkage
SYSTEM MODEL (LOGICAL) *Designer*	e.g. Logical Data Model Ent = Data Entity Reln = Data Relationship	e.g. Application Architecture Proc = Application Function I/O = User Views	e.g. Distributed System Architecture Node = I/S Function (Processor, Storage, etc) Link = Line Characteristics
TECHNOLOGY CONSTRAINED MODEL (PHYSICAL) *Builder*	e.g. Physical Data Model Ent =Segment/Table/etc. Reln =Pointer/Key/etc.	e.g. System Design Proc = Computer Function I/O = Data Elements/Sets	e.g. System Architecture Node = Hardware/Systems Software Link = Line Specifications
DETAILED REPRESEN-TATIONS (OUT-OF-CONTEXT) *Sub-Contractor*	e.g. Data Definition Ent = Field Reln = Address	e.g. Program Proc = Language Statement I/O = Control Block	e.g. Network Architecture Node = Address Link = Protocol
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK

Figure 5 : Zachman Enterprise Architecture Framework

Cette représentation matricielle du SI fait certes le bonheur des théoriciens mais dans la pratique, elle suppose une démarche séquentielle, de la définition des processus au développement des applications du SI : analyse fonctionnelle, puis élaboration exhaustive des modèles conceptuels/logiques/physiques des données et traitements, pour aboutir aux phases d'implémentation. A chaque « ligne » est associée une modélisation exhaustive des données (le « quoi »), des traitements (le « comment »), de l'implantation des données et traitements (le « où »).

Cette démarche « top-down » est réalisable en théorie avec un gigantesque outil CASE mais irréaliste sur le plan pratique dans un environnement d'entreprise, où les processus métier sont transverses à plusieurs applications : une modification du processus métier au niveau de la vue fonctionnelle du modèle devra se traduire en un changement d'implémentation logicielle au niveau d'un ensemble d'applications hétérogènes (modèles physiques de données, procédures stockées, scripts PL/SQL, transactions CICS…), le plus souvent maintenues par des équipes projets différentes…

Zachman et Merise sont des méthodes qui modélisent exhaustivement l'ensemble des Données et Traitements d'un système. Cette absence d'abstraction les limite à un périmètre beaucoup plus restreint que le SI.

Et pourtant, les méthodes Zachman et Merise ont su s'affirmer pendant près de deux décennies ! En y regardant de plus près, c'est en fait dans le domaine du développement d'applications qu'ont effectivement percé ces méthodes et non dans le domaine de l'architecture de SI : Merise par exemple fut largement utilisé - bien avant UML - pour l'analyse et la conception d'applications client-serveur, ou d'applications mainframe (3270, CICS)... En revanche, une méthodologie d'architecture de type Zachman/Merise au niveau SI suppose de définir exhaustivement des modèles conceptuels et physiques de données/traitements pour l'ensemble des applications du SI, avec toutes les contraintes techniques et organisationnelles qui en découlent…

Le flop de Merise OO…
Dans le domaine de l'ingénierie logicielle, le paradigme « Données/Traitements » et les méthodes Merise furent utilisées bien avant UML. Face aux apports du paradigme objet (abstraction, encapsulation), Merise fut rapidement abandonnée, non sans un dernier sursaut, la méthode OOM (Object Oriented Merise) : il s'agissait grossièrement de renommer « traitement » en « objet », sans remettre en cause les fondations de la démarche, centrée sur la donnée ! Cette méthode est ainsi restée bien à l'abri, dans les rayons des bibliothèques…

Quant au paradigme « Données/Traitements », il n'est pas en lui-même antinomique avec une méthodologie d'architecture de SI : on peut très bien structurer un SI en « traitements » sans pour autant tomber dans les travers de l'exhaustivité de modélisation et du syndrome « top-down », il suffira de savoir se limiter à des niveaux de modélisation suffisamment élevés, qui se concentrent sur l'essentiel des fonctionnalités. En outre, ce genre d'approche pourra même permettre la re-utilisation des traitements entre applications !

Les travers des méthodes Zachman/Merise sont donc principalement une modélisation trop exhaustive des données/traitements, utopique à l'échelle du SI ; ainsi qu'une croyance trop forte dans les démarches séquentielles, plutôt que le paradigme « Données/Traitements ». Ces méthodes ont échoué à l'échelle du SI pour ces deux raisons, et par conséquent, elles sont accusées de façon un peu trop rapide de ne pas permettre la mutualisation des traitements au niveau du SI : c'est ce point que devait résoudre la très prometteuse approche « Composants »…

Architecturer le SI : l'approche « Composants »

Avec la montée des méthodologies et langages OO au début des années 90, puis l'apparition des premiers ORB[32], un nouveau courant d'architecture de SI est apparu au milieu des années 90 : les architectures de systèmes distribués.

Surfant sur la vague objet, un nouveau courant d'architecture est apparu autour des systèmes distribués : de Corba aux serveurs de composants.

Systèmes Distribués et Composants… Ne confondons pas !
Un système distribué est un système constitué d'une plate-forme d'exécution (ORB), proposant des services techniques à des composants (accès distants, persistance, transactions, sécurité, nommage...). Les composants sont répartis sur cette plate-forme, ils implémentent la logique métier et exposent des services aux autres composants, à d'autres applications…
En dehors des environnements spécifiques ORB ou serveurs d'applications J2EE/.Net, on parle également de composants pour désigner plus généralement un module autonome exposant des services aux autres modules, sans considérer l'environnement d'exécution de ces composants ; le composant est caractérisé par ses interfaces, son comportement dynamique, son modèle de données à l'interface… Une transaction CICS unitaire ou un Web Service de type « effectuer un virement bancaire » sont à ce titre tous deux des composants.

Ce courant s'est accompagné de plusieurs méthodologies d'architecture, dont les objectifs restaient comparables aux initiatives Zachmaniennes, à savoir fournir un cadre unifié pour l'architecture de SI (du métier à l'infrastructure technique) mais s'adressant cette fois-ci aux systèmes distribués.

Le cadre normatif en la matière est le modèle RM-ODP[33] normalisé conjointement par ISO/ITU au milieu des années 90. C'est un modèle qui fut élaboré sous l'influence du framework Zachman (on y retrouve notamment cinq vues d'architecture suivant les acteurs) mais guidé par le paradigme OO et non par le paradigme Données/Traitements.

Dans les grandes lignes, RM-ODP fournit un cadre d'architecture basé sur cinq vues du système :

- **La vue Entreprise**, qui décrit les activités métier du système,
- **La vue Information,** qui définit l'information traitée par le système et la façon dont elle est traitée par les différents composants,
- **La vue Traitement**, qui spécifie fonctionnellement les traitements effectués par les différents composants
- **La vue Ingénierie** qui décrit les mécanismes logiciels permettant la distribution des composants et leur exécution sur les plates-formes d'exécution (ORB, serveur d'applications),
- **La vue Technologie** qui définit les technologies matérielles et logicielles utilisées pour l'infrastructure d'exécution, leur configuration…

Il est à noter que la vue « Traitement » s'attache à architecturer le système en composants tout en faisant abstraction des mécanismes techniques de la plate-forme d'exécution ; en ce sens, elle n'est pas spécifique aux systèmes distribués, à la différence des deux dernières vues. Dans l'absolu et en dehors du contexte RM-ODP, cette vue « Traitement » pourrait être utilisée pour architecturer un SI à base de composants qui seraient des transactions CICS, des services Web…

32 : ORB : Object Request Broker, infrastructure d'exécution d'objets répartis, cf. norme CORBA de l'OMG.
33 : Reference Model for Open Distributed Processing.

RM-ODP est un modèle de référence à partir duquel vont se dériver des méthodologies d'architecture pour les systèmes distribués.

RM-ODP n'est pas une méthodologie, c'est un modèle de référence plutôt abscons et rébarbatif qui utilise un formalisme précis et peu répandu dans les entreprises… C'est la raison pour laquelle différentes méthodologies d'architecture « allégées », inspirées de ce modèle, sont apparues. Elles utilisent un formalisme connu basé sur UML, comme par exemple Catalysis[34], ou encore des méthodologies élaborées spécifiquement par et pour certaines grandes entreprises françaises[35].

La figure suivante illustre une des phases de la démarche Catalysis qui consiste à spécifier fonctionnellement un composant (cf. vue Traitement du modèle RM-ODP).

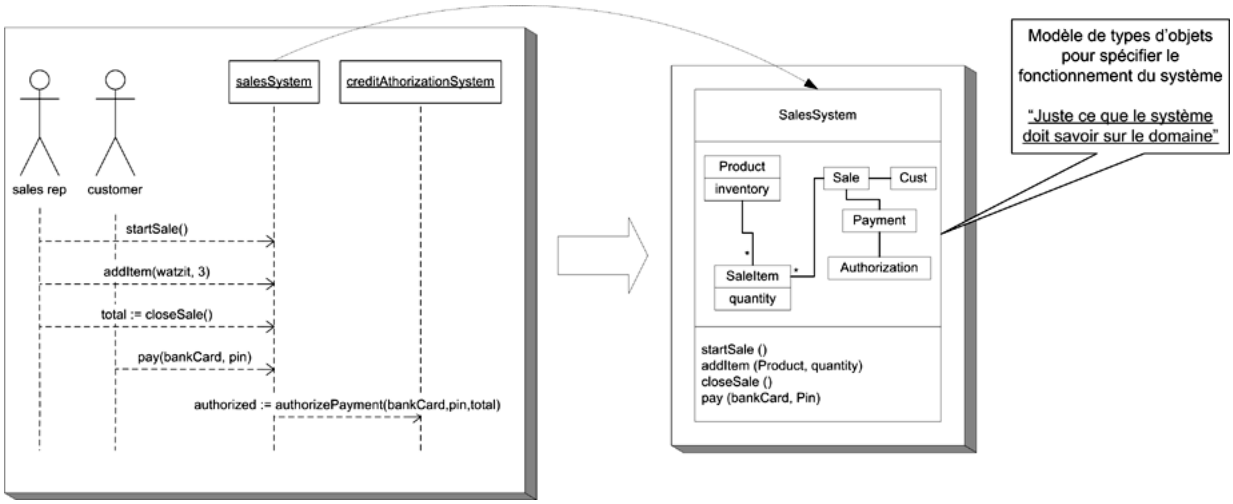


Figure 6: Illustration de la méthode Catalysis pour la spécification d'un composant dédié à la vente

Les méthodologies dérivées de RM-ODP, tout comme les méthodologies Zachman-Merise, sont des méthodologies top-down qui supposent de spécifier dans le détail les modèles de tous les niveaux (fonctionnels, techniques...)

MDA
MDA (Model Driven Architecture, littéralement Architecture guidée par les modèles), est une récente initiative en cours de standardisation à l'OMG. L'idée est de partir des modèles du domaine métier de l'entreprise (au sens UML du terme), pour définir des PIM (Platform Indépendant Model), vision abstraite d'un composant métier. Ce PIM est ensuite dérivé en différents PSM (Platform Specific Model) en fonction de l'infrastructure d'exécution : CORBA, EJB, .NET, SOAP…
L'objectif est louable, standardiser par un méta-modèle basé sur UML la représentation abstraite d'un composant et sa représentation « technique ». Mais cette initiative est encore trop jeune pour servir de base à de nouvelles méthodologies d'architecture : malheureusement, seuls les PSM CORBA sont aujourd'hui définis…

Mêmes causes, mêmes échecs : tout comme les méthodologies Zachman/Merise, les méthodologies d'architecture de SI dérivées de RM-ODP se sont limitées à des périmètres projet et n'ont pas réussi à s'étendre au niveau SI à cause de leur caractère exhaustif (spécifications détaillées à tous les niveaux) et séquentiel (top-down) - utopiste au niveau du SI. Pire encore, les méthodologies dérivées de RM-ODP supposent que le SI peut être implémenté sous la forme

34 : www.catalysis.org.
35 : France Télécom (Memento Technique N°14, www.rd.francetelecom.fr).

d'un système distribué étendu ! D'où les limites de ces méthodologies pour prendre en compte les « legacy systems », les moniteurs transactionnels autres que les OTM[36], les MOM[37], les progiciels intégrés, etc. Car ces systèmes doivent être systématiquement accédés comme des objets, mais ne le sont pas dans la plupart des cas.

En conclusion, ces méthodologies d'architecture sont intéressantes pour des projets spécifiques bâtis sur des architectures distribuées, mais non applicables au niveau du SI. En revanche, l'approche consistant au niveau de la vue « Traitement » à structurer son SI en composants, indépendamment de toute plate-forme d'exécution, reste un point fort de ces démarches.

Bilan et conclusions…

Deux écueils ont fait échouer les méthodologies d'architecture (Zachman, RM-ODP) à l'échelle du SI : dogmatisme et manque de modèles.

La méthodologie de construction du Système d'Information n'existe pas encore, mais elle correspond à un besoin criant des grandes entreprises et des entreprises en réseau.

L'Histoire nous permet de réaliser les causes d'échec de ces initiatives, que l'on peut synthétiser en deux principaux griefs :

- Le **dogmatisme** : la croyance dans une démarche top-down séquentielle (de la stratégie au code) et aux principes de projection automatique entre les niveaux.

- Le **manque de modèles** pour faire converger les réflexions. Car en dehors de leurs spécificités stratégiques, deux banques ou deux opérateurs auront des systèmes qui finalement devront se ressembler, car ils évoluent dans les mêmes réseaux d'interopérabilité.

Evitons les erreurs du passé en proposant une méthodologie d'architecture pragmatique, basée sur un formalisme simple et des patterns.

La méthodologie d'architecture de SI que nous proposons tente de gommer ces deux défauts par une emphase particulière sur :

- Le pragmatisme :
  - une méthode centrée sur **l'itération** et l'interactivité entre fonctionnels et techniques,
  - une **rupture assumée** mais maîtrisée entre architecture générale et conception détaillée,

- Le formalisme et les modèles :
  - Les modèles utiles à la conception sont identifiés, avec une responsabilité claire, sous un formalisme au **standard UML**,
  - La constitution d'un portefeuille de **patterns fonctionnels** permet de restreindre le champ des possibilités et de converger vers des solutions éprouvées.

36 : OTM : Object Transaction Monitor, moniteur transactionnel objet (utilisé dans les architectures ORB).
37 : MOM : Message Oriented Middleware.

IV. Les outils de l'architecte

Ce chapitre se propose d'expliciter ce que nous entendons par **formalisme** pour une architecture de SI, et d'autre part de préciser la notion de **modèle**, ou pattern d'architecture.

Le formalisme de modélisation

Le besoin d'un formalisme de modélisation d'architecture

Le chapitre précédent concluait sur le besoin d'un formalisme et de modèles adaptés à la modélisation d'architecture de SI. Il ne se dégage en effet pas encore de consensus sur ce sujet, et chaque outil amène généralement son propre formalisme de modélisation (MEGA, Amarco, etc.).

Nous proposons d'appuyer notre formalisme de modélisation sur UML, qui est le standard aujourd'hui largement accepté dans le monde de l'architecture logicielle. Nous avons sélectionné les diagrammes qui nous semblent adaptés à la modélisation d'architecture de système d'information. Notre approche restant pragmatique, nous avons donc privilégié la clarté et l'utilité des modèles au respect strict des règles d'UML.

Point de vue et redondance d'information

Certains diagrammes UML peuvent sembler redondants mais cette redondance fait partie du formalisme UML. Par exemple, le diagramme de séquence et le diagramme de collaboration sont intimement liés. Il est toutefois intéressant de changer de point de vue pour éclaircir l'architecture et/ou en faire ressortir une opportunité d'amélioration.

Chaque diagramme permet de visualiser l'architecture suivant un angle différent et permet donc de détecter des erreurs ou de proposer des modifications. Par exemple, un diagramme de collaboration montrera mieux qu'un diagramme de séquence qu'une brique applicative échange de nombreux flux avec le système, ce qui poussera peut-être l'architecte à modifier l'architecture afin de rationaliser les échanges.

Note sur les outils de modélisation : plus qu'un outil, c'est l'utilisation d'un formalisme de modélisation qui importe. Les exemples de diagramme UML qui suivent, ont été réalisés avec Rational Rose. Il aurait été possible d'utiliser d'autres outils tels que MEGA, Visio ou TogetherJ.

Les diagrammes fonctionnels

Les équipes fonctionnelles produisent et ont la responsabilité de diagrammes que nous regrouperons dans un « volet fonctionnel » :

- Diagramme d'organisation,
- Diagramme de cas d'utilisation,
- Diagramme d'activité,
- Diagramme d'état.

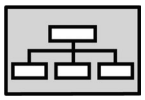


Diagramme d'organisation

Le diagramme d'organisation n'est pas un diagramme UML mais il nous semble très important car il n'est pas possible de s'affranchir de l'organisation d'une entreprise pour concevoir son SI.

Le diagramme d'organisation permet de donner une vision générale de l'organisation de l'entreprise sur le périmètre du projet.

Prenons l'exemple de la mise en place d'un système de gestion de production. Les principales directions utilisatrices de ce système sont : la direction commerciale, la direction de la production et la direction des achats. Les autres départements tels que les ressources humaines, la finance et l'informatique sont hors périmètre pour ce projet car elles n'utilisent pas directement le système.

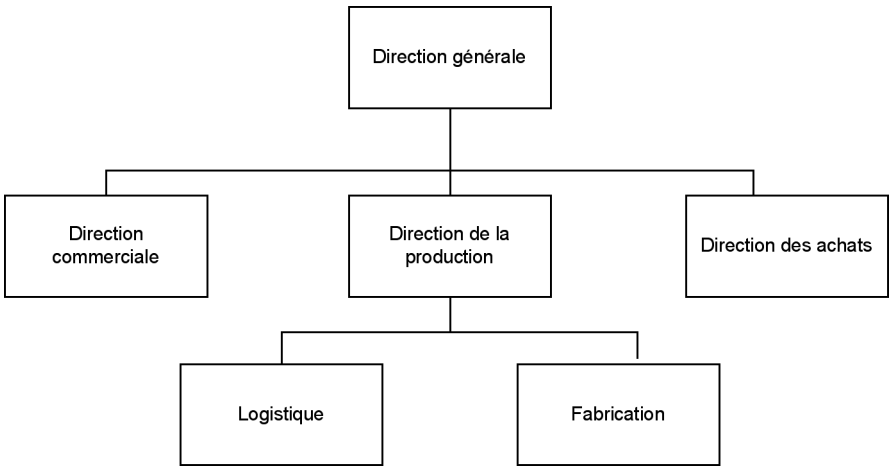


Figure 7 : Diagramme d'organisation restreint au périmètre projet

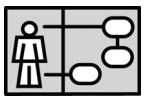


Diagramme de cas d'utilisation

Un cas d'utilisation représente une interaction entre un ou des acteur(s) et le système d'information. Il permet d'exprimer les services rendus par le système. Un diagramme de cas d'utilisation regroupe plusieurs cas d'utilisation qui ont un fort lien fonctionnel.

L'intérêt d'un tel diagramme est d'avoir du sens pour tous les acteurs du projet et notamment les utilisateurs. La bonne idée des cas d'utilisation est de décrire le système à travers des interactions avec ses acteurs. On ne se situe pas sur des modèles logiques issues de réflexions théoriques, mais bien sur des modèles d'utilisation réelle, compréhensibles par tous.

Voici un exemple de diagramme qui regroupe les principaux cas d'utilisation liés à la direction commerciale :

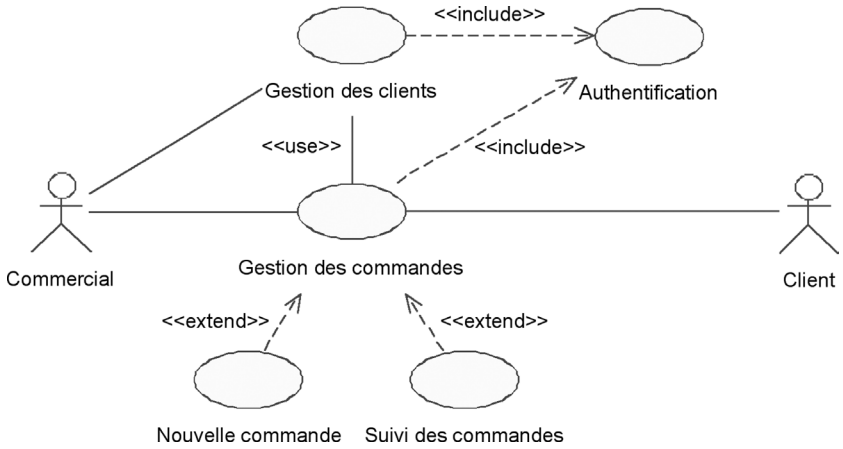


Figure 8 : Diagramme de cas d'utilisation liés à la direction commerciale

Nous n'avons considéré que les acteurs liés à l'usage normal du système et non des acteurs exceptionnels comme un « gestionnaire ». La réflexion sur les cas d'utilisation doit être cohérente avec le périmètre du projet. Un travers classique consiste à vouloir décrire trop finement les cas d'utilisation, avec le risque de dériver vers le « comment » au lieu de rester dans le « quoi » à ce stade.

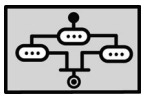


Diagramme d'activité

Un diagramme d'activité décrit l'enchaînement des activités participant à la réalisation d'un cas d'utilisation (UC pour use-case). Un diagramme d'activité intègre tous les scénarios d'exécution possibles d'un cas d'utilisation et fait donc apparaître des embranchements, des choix conditionnels et des rendez-vous.

Le point de vue reste celui de l'acteur principal du cas d'utilisation et il ne s'agit donc pas de décrire les activités internes du système mais bien celle de l'acteur.

Ce diagramme permet de détailler visuellement un cas d'utilisation. Tous les cas d'utilisation ne doivent pas obligatoirement être décrit par un diagramme d'activité. Par exemple, le UC « Authentification » qui est inclus dans les UC « Gestion des clients » et « Gestion des commandes », n'a probablement pas besoin d'être détaillé par un diagramme d'activité.

L'exemple suivant décrit le diagramme d'activité associé au cas d'utilisation « Nouvelle commande » :

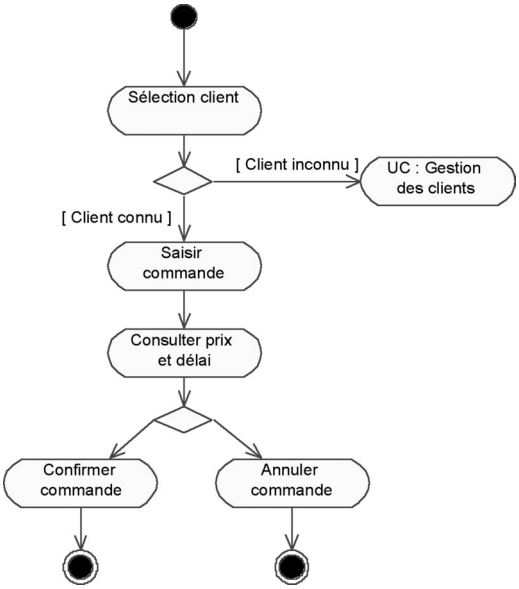


Figure 9 : Diagramme d'activité du cas d'utilisation « Nouvelle commande »

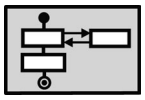


Diagramme d'état des objets métiers

Un diagramme d'état décrit le cycle de vie d'un objet métier manipulé par le système.

Les objets métier auxquels l'architecte s'intéresse sont ceux qui ont un sens pour les équipes fonctionnelles. On ne cherchera pas à « deviner » les objets internes du système, et on s'intéressera donc à ceux exprimés par les experts fonctionnels. Pour certains objets métier, détailler le cycle de vie permet d'enrichir la réflexion, mais il faut bien les choisir. Dans notre exemple, nous identifions les objets Commande, Client, Ordre de Fabrication, Bon de Livraison, etc.

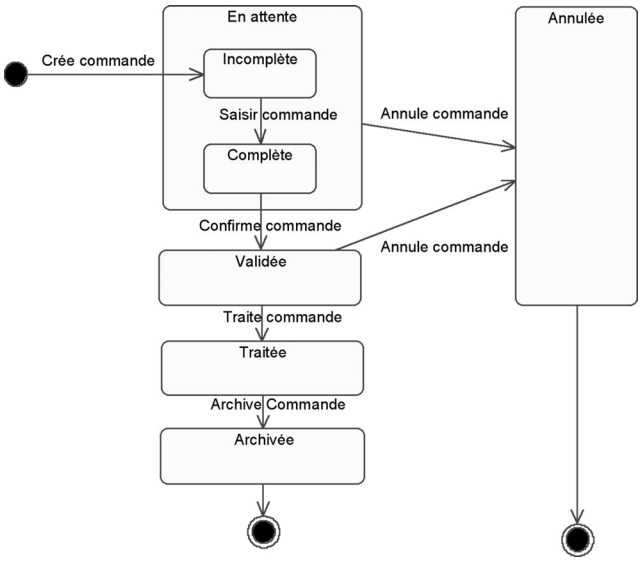


Figure 10 : Diagramme d'état de l'objet « Commande »

Les diagrammes applicatifs

Les diagrammes du volet applicatif sont sous la responsabilité de l'équipe d'architecture :

- Diagramme de briques applicatives,
- Diagramme de séquence,
- Diagramme de collaborations.

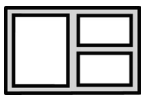


Diagramme de briques applicatives

Ce diagramme est utilisé pour décrire le découpage système en briques applicatives.

Il est basé sur un diagramme de classes d'UML et utilise un stéréotype « Brique ». On appelle « Brique » un sous-système applicatif associé à un niveau de détail donné. Il est possible de zoomer dans une brique et de montrer les sous-briques qui la constitue.

Voici un exemple de « diagramme de briques applicatives » :

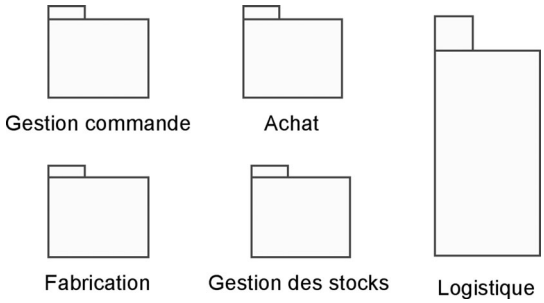


Figure 11 : Diagramme de briques applicatives

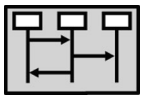


Diagramme de séquence

Un diagramme de séquence décrit les interactions entre les acteurs et les briques applicatives du système dans le contexte du déroulement d'un scénario d'un cas d'utilisation.

Une fois les briques applicatives définies, il est possible d'expliciter le déroulement des cas d'utilisation sur le système. L'objectif est de s'assurer que les cas d'utilisation sont réalisables avec notre modèle de briques applicatives.

Voici le diagramme associé au cas d'utilisation « Nouvelle commande » pour le cas d'une commande dont tous les articles sont en stock :

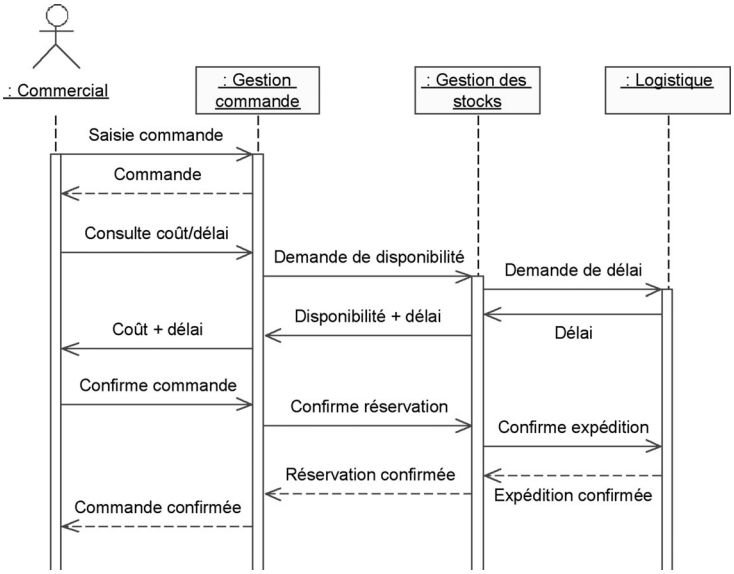


Figure 12 : Diagramme de séquence pour le passage d'une nouvelle commande

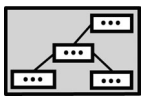


Diagramme de collaboration

Un diagramme de collaboration représente les échanges et les mécanismes de coopération entre les différentes briques du système.

Ce diagramme peut contenir des informations redondantes avec le diagramme de séquence. D'ailleurs certains outils permettent un passage presque automatique de l'un vers l'autre. Son atout est de présenter les échanges d'un point de vue flux, alors que le diagramme de séquence s'attache à en présenter le séquencement.

Dans notre exemple :

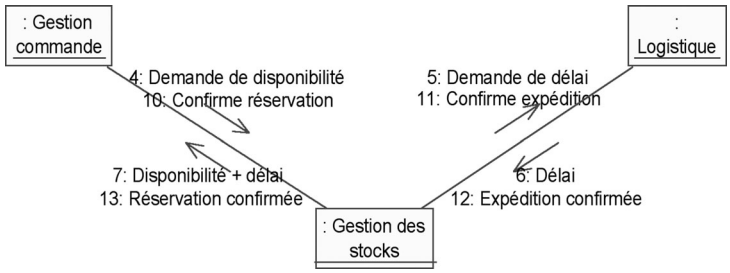


Figure 13 : Diagramme de collaboration

Synthèse

Volet fonctionnel	
Type	**Objectifs**
Diagramme d'organisation	Conserver une cohérence entre SI et organisation
Diagramme de cas d'utilisation	Décomposer et regrouper les cas d'utilisation par affinité fonctionnelle
Diagramme d'activité	Zoomer sur les activités réalisées dans certains cas d'utilisation
Diagramme d'état	Décrire les cycles de vie complexes des principaux objets métiers

Volet Applicatif	
Type	**Objectifs**
Diagramme de briques applicatives	Présenter le découpage du SI en briques applicatives
Diagramme de séquence	Valider l'architecture en déroulant les cas d'utilisation sur les briques applicatives
Diagramme de collaborations	Mettre en évidence les flux qu'échange une brique applicative avec le reste du système.

Nous avons vu que la communication est un des enjeux majeurs de tout projet informatique d'envergure. L'utilisation du formalisme de modélisation graphique que nous venons de décrire aide l'architecte à établir un langage commun entre les différents acteurs du projet.

Des patterns pour structurer la conception

Un pattern est une manière de résoudre un problème récurrent, c'est un modèle. Par exemple, on utilise le pattern « Modèle Vue Contrôleur » (MVC) pour bâtir des interfaces homme/machine complexes. Ce pattern impose une séparation entre le Modèle IHM (la forme des écrans et des états), la Vue qu'il donne sur les données, et le Contrôleur qui régit les interactions entre écrans. C'est un standard, aujourd'hui incorporé dans la plupart des outils, et qui a contribué à l'amélioration notoire de la productivité et de la qualité des développements d'IHM.

Le pattern fonctionnel s'intéresse autant à la future organisation des utilisateurs du SI, qu'à la structure du SI lui-même.

En architecture de Systèmes d'Information, le pattern aura également ce rôle de structuration. Il est d'autant plus pertinent que le problème posé est nouveau, et donc que le nombre d'inconnues est important. L'application d'un pattern va conduire à réduire le champ des possibilités.
Le pattern d'architecture s'intéresse autant à l'organisation du SI (son architecture), qu'à l'organisation des utilisateurs du SI (nous le verrons plus loin). Mais le lecteur attentif aura réagi à ce stade : beaucoup de dirigeants nous demandent en effet de réaliser des Systèmes d'Information indépendants de l'organisation... C'est une pure utopie. Un SI est précisément au service d'une organisation. Si l'on souhaite qu'il serve les besoins d'une organisation différente, on devra soit le modifier, soit avoir prévu en amont ce changement, et payer les coûts de cette flexibilité.
Prenons un exemple simple, le cas de la relation client. Si un SI est organisé selon des lignes « produits », il ne sera pas adapté à une gestion transverse du client. Si l'organisation change avec la mise en place de chargés de clientèle multi-produits, il faudra modifier le SI en conséquence pour faire apparaître un outil utile à cette nouvelle population d'utilisateurs.

Nous le rappelons, les technologies de l'Information sont des technologies au service de l'Organisation.

Au niveau des patterns, nous distinguerons deux catégories :

Les patterns techniques permettent quant à eux de zoomer sur des problèmes informatiques purs.

- les **patterns fonctionnels**, qui ont un impact sur l'organisation et la structuration du SI, comme par exemple la notion de Royaume/Emissaire ou de Noyau.
- les **patterns techniques**, qui résolvent des problèmes d'architecture à un niveau de maille plus fin, comme par exemple le pattern Single Sign On ou Datawarehouse.

Le pattern fonctionnel Royaume-Emissaire

Initialement imaginé par les usines de développement logiciel de Microsoft, sous la houlette de Pat Helland, la notion de Royaume et d'Emissaire a été utilisée pour développer des systèmes de bases de données complexes comme le serveur de messagerie Exchange. L'idée repose sur le constat que tout système est composé d'un cœur (le Royaume) et de fonctions dédiées à la communication extérieure (l'Emissaire), et que ces fonctions gagnent à être séparées. Le royaume ne fait confiance à personne, et ne communique qu'avec son émissaire. L'émissaire a donc la responsabilité de filtrer et transformer les sollicitations extérieures avant de transmettre son travail au royaume.

Un Royaume pour le cœur de métier, un ou des Emissaires pour les activités liées au monde extérieur.

Le pattern Royaume/Emissaire s'applique à un niveau Système d'Information, et il impose une organisation adaptée. Imaginons par exemple, le cas du système d'information opérant le fichier d'empreintes ADN national. Ce système est constitué d'un Royaume dont l'objet est la gestion courante des données d'empreintes génétiques : ajout et modification d'éléments dans la base, recherche multicritères, facturation, etc. L'Emissaire du système aura lui pour rôle d'assurer que

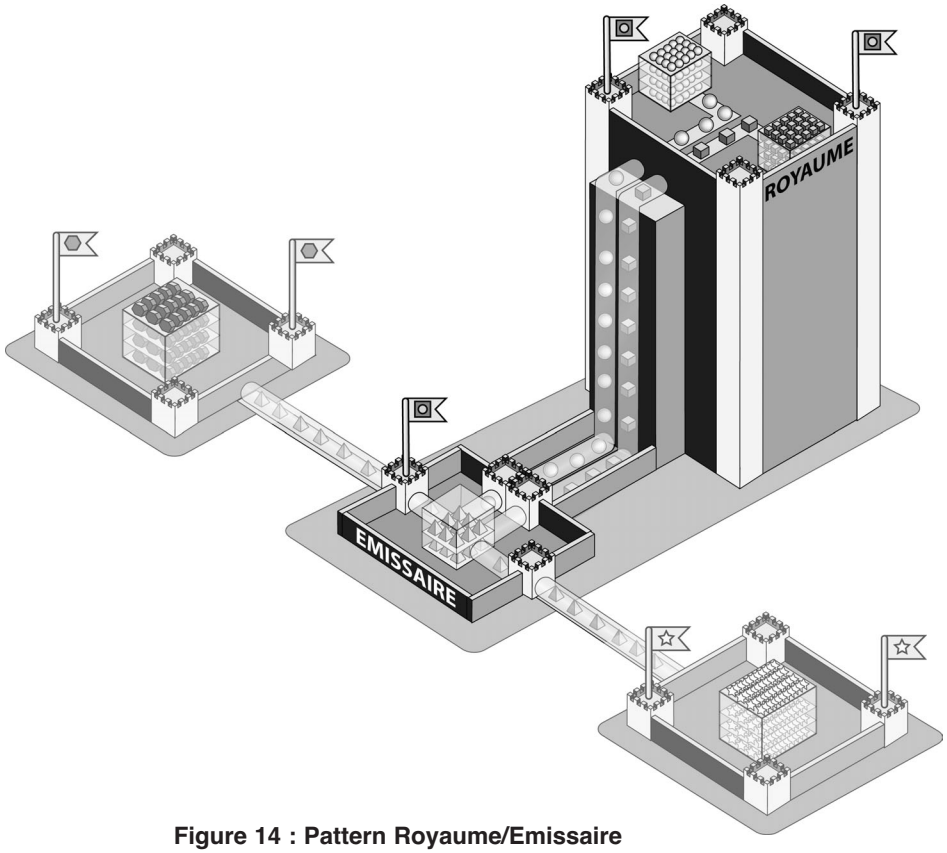


Figure 14 : Pattern Royaume/Emissaire

les requêtes utilisateur sont bien conformes à la politique d'utilisation du système. Par exemple, seuls les enquêteurs habilités et munis d'un mandat spécifique relatif à une enquête seront autorisés par l'Emissaire à accéder au royaume. Dans notre exemple, si le nombre d'accès au fichier est faible, l'Emissaire peut même n'avoir aucune existence informatique, pour ne se matérialiser qu'en du personnel chargé de valider les requêtes des polices nationales ou internationales.

Le pattern Royaume/Emissaire est toujours pertinent lorsqu'un système a de fortes contraintes de sécurité, mais il l'est aussi lorsque ce système requiert des interactions nombreuses et variées avec l'extérieur.

De manière synthétique, l'émissaire contiendra tout élément susceptible de remplir les fonctions suivantes :

- **publier les services et processus offerts par canaux**, canaux étant pris au sens large : canaux physiques (Web, mail, fax, téléphone…) et logiques (employés, partenaires, revendeurs...). L'Emissaire prend en charge les spécificités des processus de chaque canal pour les masquer au Royaume.

- **contrôler et habiliter les sollicitations extérieures**, vérifier l'identité, la signature, la confidentialité, la non-répudiation[38], le droit d'accès. Cette dernière fonction étant souvent la plus complexe.

38 : Voir Le Livre Blanc de la Sécurité – © OCTO Technology 2001.

- **réconcilier les visions extérieures et intérieures**, car l'émissaire aura en sa possession un certain nombre de données de référence, dont il diffusera des clichés vers ces utilisateurs. Par exemple dans le cas d'un émissaire « Catalogue papier » d'un Royaume de « vente par correspondance », les utilisateurs disposent du cliché des prix en date d'édition du catalogue, ceux-ci ont pu varier dans le Royaume, qui lui, fait référence.

L'Emissaire décharge le Royaume des fonctions non vitales, il exploite donc des objets métier inconnus du Royaume.

D'un point de vue technique, **l'Emissaire va nécessairement manipuler de nouveaux objets métier** (voir paragraphe « Formalisme de modélisation »), non disponibles dans le Royaume. Dans notre exemple, il s'agira en particulier du dossier de requête, composé de la ou des requête(s) au fichier et des pièces justificatives. L'Emissaire utilise le dossier pour filtrer les requêtes, le Royaume ne s'intéresse qu'aux requêtes elles-mêmes.

Enfin, il est généralement pertinent d'utiliser un formalisme pivot unique entre le royaume et l'émissaire (ici les requêtes), l'émissaire se chargeant des conversions entre ce format et les différents formats de canaux qu'il gère, dont ceux manuels. XML est bien entendu le choix technique préféré pour prendre en charge cette sémantique.

Le pattern fonctionnel NOYAU

Straight Through Processing, une discipline pratiquée dans tous les secteurs d'activité.

Le noyau est un pattern que nous allons utiliser dans les environnements où le rôle du système d'information est d'automatiser des tâches volumineuses et impliquant de nombreux systèmes. C'est en particulier le cas dans certains secteurs de la finance, où le Straight Through Processing (STP), c'est-à-dire la capacité à traiter une instruction sans intervention humaine, est fondamentale. Augmenter le taux de STP de quelques points peut générer des gains de productivité colossaux du fait des volumes traités. Le STP est une problématique que l'on retrouve plus largement chez les opérateurs téléphoniques (facturation, roaming, compensation), la logistique dans la distribution ou la fabrication, les systèmes de réservation…

Le noyau est donc un **routeur d'évènements métier**. Quelle différence avec un outil EAI me demanderiez-vous ? La même qu'entre SAP et une base de données. Le noyau pourra être implémenté à l'aide d'un produit EAI, mais c'est avant tout une brique fonctionnelle du SI, qui aura ces utilisateurs (les « responsables STP ») et sa logique propre.

Le STP conduit à l'automatisation de processus complexes, qui impose une intégration de haut niveau entre systèmes informatiques.

Prenons l'exemple d'une problématique STP chez un conservateur de titres, c'est-à-dire l'acteur responsable de la « livraison » des produits financiers une fois leur négociation effectuée, et leur « conservation » dans un coffre multi-instruments. Le noyau va devoir manipuler les instructions des clients (ordres d'achat ou de vente sur un instrument financier particulier : cash, valeurs mobilières, dérivés, change, etc.), et les router vers le bon destinataire du traitement : traitement du cash, des valeurs mobilières, du change, etc. Jusqu'ici rien d'exceptionnel. La complexité intervient lorsque les interactions entre ces différentes briques de traitement s'intensifient, c'est-à-dire lorsque l'état d'un objet métier A dans une brique X a un impact sur ce que devrait faire l'objet B dans une brique Y. Elles dépendent très souvent du statut des objets dans chacune de ces briques. Par exemple, une instruction titres à régler en dollars va nécessiter la création d'une instruction de change pour poursuivre son dénouement. L'instruction de change va suivre son propre processus, et c'est au moment où elle va atteindre un statut particulier, disons « contrepartie trouvée », que l'instruction titre initiale va pouvoir poursuivre son propre processus. Le noyau devra comprendre l'avancement de ces différents statuts, pour réagir sur une transition de statut, en prenant en compte un historique d'instructions liées.

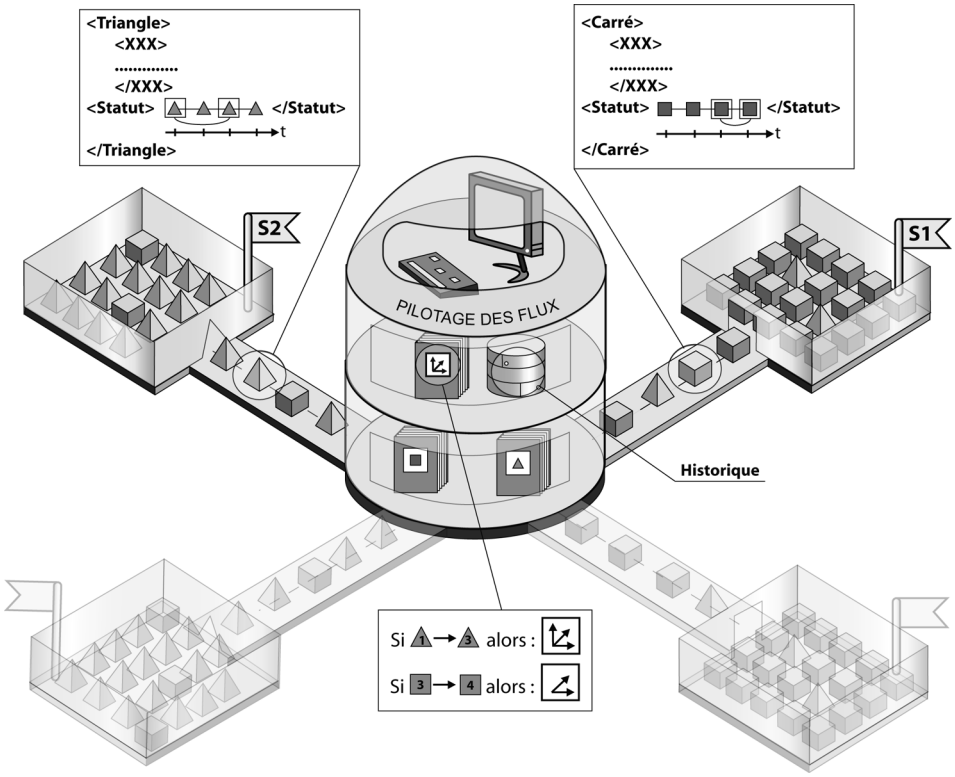


Figure 15 : Pattern Noyau

Cet exemple nous illustre les concepts manipulés par le noyau :

- le noyau comprend les **objets métier** qui le traversent : il a donc un dictionnaire de ces objets, et il en connaît les attributs spécifiques qui vont lui permettre de prendre des décisions. Il définit la norme de ces attributs, c'est-à-dire l'ensemble des valeurs qu'ils peuvent prendre. Dans notre exemple : « instruction prise en charge », « contrôlée », « émise sur les marchés », « appariée[39] », « dénouée » sont les statuts nominaux, auxquels on devra adjoindre des états liés aux erreurs et aux annulations (« suspens », « annulée »).

- le noyau notifie ces collatéraux des évènements métier, sur la base d'une **table de routage statique ou dynamique**

  - **statique** dans le cas où les règles métier s'appliquent sans que le noyau n'aie à se souvenir du passé (noyau sans état)

  - **dynamique** dans le cas où les règles métier s'appliquent également sur la foi du passé. Dans notre exemple il s'agira d'une base de liens entre instructions subordonnées.

39 : La contrepartie a été trouvée.

Pour créer un noyau, il faudra donc s'intéresser :

- au **format pivot** des objets métier transverses au SI, et à leur transformation vers les formats supportés par les collatéraux[40],
- aux éléments sémantiques transverses à tout le SI, et notamment la notion de statut,
- aux données qui doivent persister dans le noyau pour garantir son bon fonctionnement. Il n'est pas rare que le noyau finisse par alimenter tous les systèmes transverses (infocentres facturation, rentabilité, piste d'audit, risques, SLA...), sans pour autant posséder cette donnée lui-même.
- aux **fonctions qu'il doit présenter à ses utilisateurs** : réparation d'objet métier, rejeu d'objet métier, annulation de processus, statistiques, etc.
- au caractère **actif ou passif** : le noyau a-t-il une copie des données de processus présentes dans les collatéraux, copies qu'il consolide en une vision globale, ou bien crée-t-il lui-même du processus inexistant ailleurs et possède-t-il en propre des données de processus ?

Etre très vigilant sur le lieu de vie des processus de l'entreprise : processus front-office, processus STP de back-office ou nouveaux processus transverses ? Quelle responsabilité pour ces derniers ?

Noyau et Business Process Management (BPM)

Beaucoup a été écrit sur le Business Process Management, nouvelle idole technologique supposée créer des processus dans toute l'entreprise. Intéressons-nous à la réalité des faits. Dans un Système d'Information classique, nous allons constater que les processus de vente sont gérés par le SI Front-Office, par exemple un progiciel de Gestion de la Relation Client, tandis que les processus de back-office sont gérés par l'ERP. Il est des cas moins manichéens bien sûr, mais l'intérêt de l'exemple est de montrer que des processus vivent déjà dans des parties du SI de l'entreprise. **La question à se poser en architecture est de savoir si l'intégration entre deux domaines crée du processus supplémentaire ou pas**. Et c'est loin d'être toujours le cas, n'inventons pas du BPM là où il n'y en a pas. Dans ces situations, un noyau statique (sans état), voire une intégration point à point permettront d'assurer l'interopérabilité des domaines.

Lorsque l'orchestration de fonctions nécessite un état, comme par exemple le cas de notre noyau STP financier, nous créons de fait du processus. Ceux-ci peuvent être du domaine front-office, donc visibles du client (comme le déclenchement automatique d'ordre sur atteinte de seuils), ou du domaine back-office, avec l'automatisation de processus internes (le cas de notre instruction de change, d'un appel de marge automatique, etc.). Ainsi la question fondamentale à se poser est de savoir à qui appartiennent les processus que nous créons, c'est-à-dire qui en sera responsable une fois en production ? Puis comment implémenter ces processus ? Dans mes progiciels métier, ou à l'aide d'un outil de BPM généraliste ? Tous les cas se présentent, mais si les progiciels de BPM peuvent aider à implémenter un noyau, ils sont souvent trop rigides pour gérer des processus dynamiques sans requérir de développement spécifique.

En conclusion, soit nous trouvons des processus métier clé en main dans des progiciels (vente, approvisionnement, etc.), soit nous devrons développer du spécifique. Ne croyons pas au mythe du gestionnaire de processus généraliste clé en main, mais recherchons plutôt des boîtes à outils de haut niveau auprès de ce type d'éditeur.

40 : Ici l'EAI peut être un facilitateur.

Le pattern fonctionnel Référentiel

Dernier pattern fonctionnel, mais non des moindres, le Référentiel. Passons rapidement sur quelques définitions : un référentiel est un réceptacle de données plutôt stables, et utilisées par au moins deux systèmes distincts, c'est donc un système mutualisé. Dans les entreprises, on distingue en particulier les référentiels employés (pages blanches internes, annuaires des utilisateurs du SI, annuaires de messagerie, etc.), les référentiels clients (bases prospects, bases clients etc.) et les référentiels produits (catalogue de produits, référentiel valeurs mobilières, prestations de service, etc.).

Mettre en place un référentiel pour réduire les coûts ou augmenter la cohérence, souvent véhicule de nouvelles opportunités.

La création d'un référentiel procède de deux volontés distinctes qui sont :

- les **économies**, par la mutualisation de systèmes autrefois distincts (ex. : référentiel valeurs)
- les **gains en cohérence**, qui peuvent être la source d'augmentation de la compétitivité (référentiel client) ou d'amélioration de la sécurité (référentiel employés).

Pourquoi un pattern ? Quelque soit la nature du problème, la création d'un référentiel se fait souvent par fusion. Fusion de systèmes, mais aussi de personnel chargé de ces systèmes. On ne crée pas un référentiel employés sans modifier profondément les processus d'inscription dans les branches de l'entreprise : ressources humaines, systèmes informatiques, téléphonie, partenaires, administrations, etc. Ne parlons pas des référentiels client, qui sont la cause même des grandes métamorphoses que subissent les entreprises actuellement.

Bien qu'on ait longtemps cru que des technologies, comme LDAP par exemple, allaient nous permettre de créer des référentiels les yeux fermés et les mains dans les moufles. L'absence de système global, automatisé et traçable de gestion des employés dans les entreprises aujourd'hui nous pousse à croire que la technique ne doit probablement pas être l'unique problème.
De même, beaucoup de théoriciens de l'Urbanisme nous proposent des beautés lyriques du type « toute donnée appartient à un bloc et un seul… ». Dans les faits, la présence d'un existant non urbanisé impose une gymnastique beaucoup plus acrobatique avec ces données distribuées dans l'entreprise ! La problématique est synthétisée dans la figure suivante :

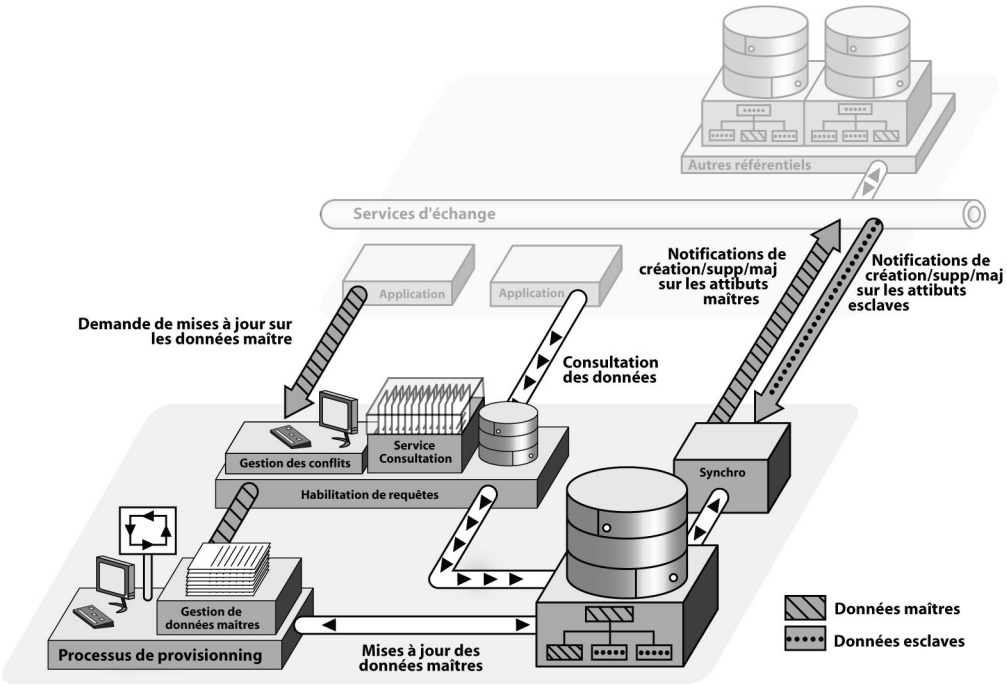


Figure 16 : Pattern Référentiel

Partager des données dans un environnement distribué et multi-maître : un problème plus compliqué qu'on le croit.

Le domaine référentiel X est responsable bien sûr de la gestion de ses données dites « maîtres ». L'existence de données maîtres présuppose celle de données « esclaves », que l'on retrouvera donc dans les systèmes utilisant cette donnée pour leurs traitements, mais n'étant pas habilités à la modifier directement. Et vice versa. Pour des raisons historiques, la plupart des Systèmes d'Information hébergent de nombreux référentiels, conduisant à un environnement multi-maître : plusieurs systèmes peuvent être « responsables » de la même donnée.

Pour assurer sa mission de gestion de données pour le compte d'autres applications, le référentiel va devoir :

- recevoir, autoriser et répondre aux accès en lecture à ses données,
- recevoir, autoriser et orchestrer les processus distribués de gestion de ses données (provisioning). C'est à ce niveau que sera géré le fait que lorsqu'un employé est embauché, on doit traiter son dossier dans différents services, comme les RH ou les services informatiques,
- propager automatiquement ces données maîtres vers l'extérieur.

Un dictionnaire de données (ou méta-données), décrit la structure des données gérées, leur propriétaire, leur cycle de vie, leurs habilitations, ou leurs cibles de réplication. Il permet aux briques de synchronisation et de gestion des données maîtres de réaliser leur travail. Ce dictionnaire gagne à être partagé, mais peut être disjoint pour des questions techniques ou liées à l'existant.

Pour conclure cette partie sur les patterns fonctionnels, notons que le pattern procède lui-même d'une **logique d'assemblage**. Et nous voyons par exemple que le cas d'un important système d'information dédié à la gestion d'un référentiel, par exemple le fichier ADN, nécessitera de s'appliquer à lui-même un pattern royaume/émissaire pour isoler toutes les fonctions d'habilitation et de synchronisation dans un système dédié, porteur d'une lourde complexité.

Enfin, le pattern doit être flexible, inutile de systématiser tout à l'infini. L'objectif est de structurer le neuf, pas de changer l'existant à tout prix. Inutile par exemple de vouloir migrer des existants dans des schémas aussi complets si cela ne crée pas spécifiquement de valeur ou d'économies.

Les patterns techniques

Il arrive souvent qu'un point du dossier d'architecture générale nécessite un zoom technique pour être mieux qualifié, nous pourrons utiliser des patterns techniques pour proposer des solutions d'architecture technique.

Des patterns techniques pour étayer un problème informatique spécifique.

Un pattern d'architecture technique pourra résoudre un problème informatique pur. Il n'a a priori pas d'incidence sur l'organisation, hormis l'exploitation du système. Le pattern Single Sign On (SSO) permet par exemple de gérer une plate-forme technique d'accès sécurisés et transparents aux applications, voire à certaines fonctions unitaires de l'entreprise. Ce pattern fera parti d'un projet portail par exemple.

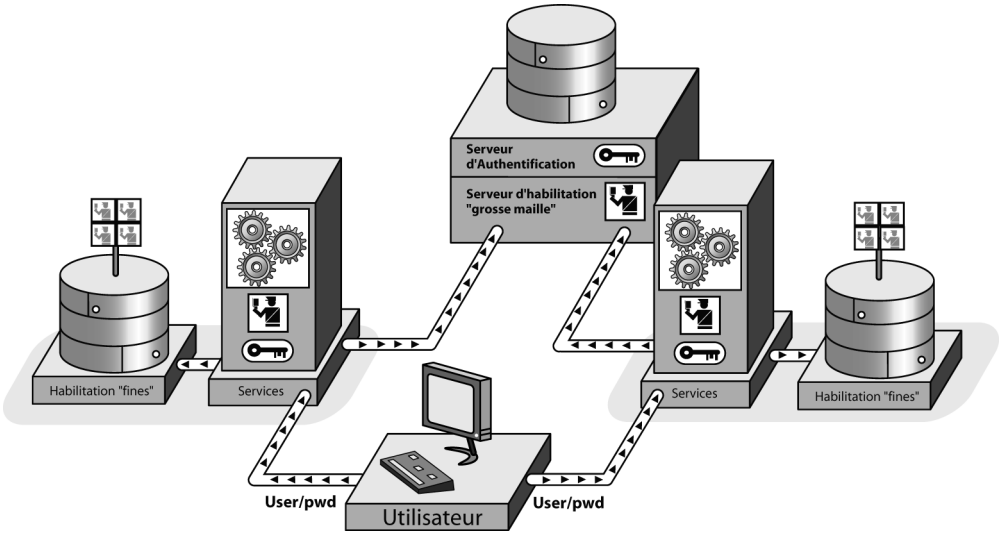


Figure 17 : Pattern SSO décentralisé, les composants de sécurité sont implantés dans chaque application

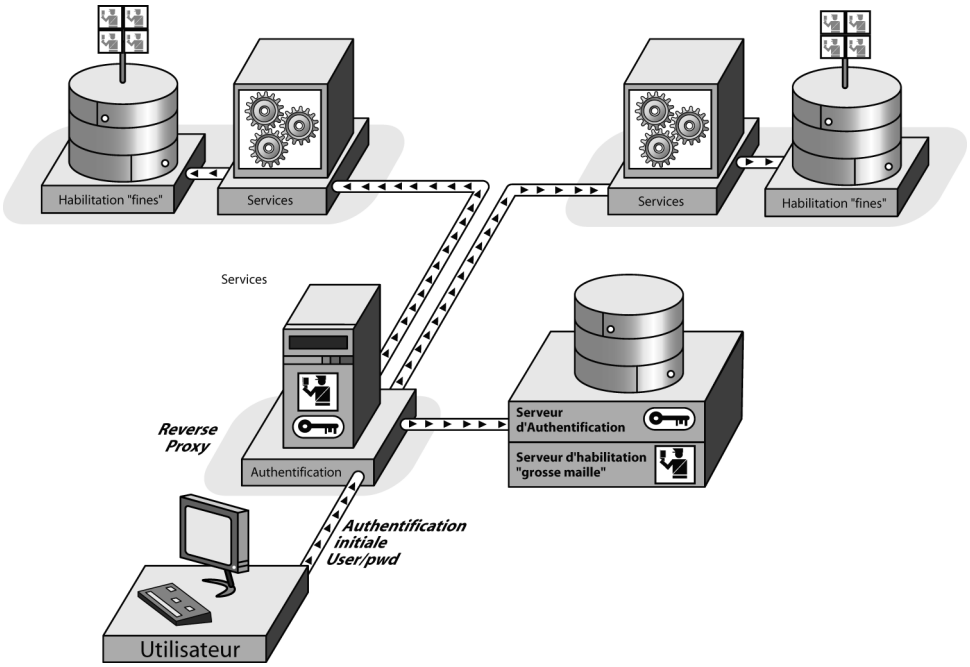


Figure 18 : Pattern SSO centralisé, les composants de sécurité sont implantés dans un Reverse Proxy unique, adapté aux architectures Web

Les patterns d'architecture technique sont nombreux, nous en trouverons autour des thèmes de la sécurité comme notre exemple, des interfaces homme/machine (architecture multi-canal, éditique …), des services métier (gestion du transactionnel, composants avec ou sans état …), de la persistance (stockage relationnel, stockage XML, stockage infocentre..), des échanges (pattern EAI ou ETL, format pivot…).
Nous ne visons pas l'exhaustivité dans cette publication, puisque ces éléments d'architecture technique sont abondamment décrits dans nos différentes publications (voir Bibliographie).

V. Démarche d'architecture projet

Eclater une problématique en plusieurs projets et réintégrer ensuite les réalisations dans une solution globale.

La complexité des projets informatiques est proportionnelle aux ambitions d'automatisation et d'optimisation des processus de l'entreprise. Ces projets sont d'autant plus complexes qu'ils doivent intégrer un existant lourd et s'insérer dans un environnement impliquant un grand nombre d'acteurs et de systèmes hétérogènes.

La difficulté relève de la capacité à concevoir une solution réalisable dans les temps et coûts impartis, sur la base de concepts simples, et à la construire sans dévier des objectifs initialement fixés. La démarche consiste en un découpage en sous-ensembles de complexité moindre donnant lieu à des chantiers de taille raisonnable et pilotable, c'est à dire compris entre 5 et 10 personnes en régime de croisière. Le défi reste ensuite de garantir l'intégration entre ces modules pour aboutir à une solution globale satisfaisant l'ensemble des besoins exprimés.

Cette démarche n'est bien sûr pas systématique. Elle n'est justifiée que dans la mesure où le projet adresse une complexité importante en termes de processus métier, de nombre de briques applicatives à combiner, ou d'environnements techniques hétérogènes à intégrer. Elle devient indispensable lorsque plusieurs générations technologiques et des progiciels divers doivent coexister.

La démarche d'architecture n'a de pertinence que dans des environnements complexes ou non-standard.

Lorsque la problématique adresse un périmètre suffisamment délimité et maîtrisé avec un minimum d'interactions avec des composants externes, le besoin d'architecture est moins évident. Il suffit de rechercher un progiciel du marché, même de niche, dont la couverture fonctionnelle serait proche du besoin ou d'envisager un développement maison. Dans ce cas-là, des démarches de type XP ou RUP[41] sont directement adaptées.

> Dans le cadre du bâtiment, on fait rarement appel à un architecte lorsqu'il s'agit de monter des maisons en préfabriquées. Le montage est généralement plutôt directement pris en charge par le fabricant ou son distributeur. Encore moins lorsqu'il s'agit de faire un ravalement de façade, ou seul le respect de certaines normes techniques, environnementales ou d'urbanisme importe. La valeur ajoutée de l'architecte se positionne pour des problèmes complexes ou non standards.

L'éthique projet prend ses racines dans un objectif clair et fondé.

En revanche, l'adoption d'une approche architecture devient indispensable lorsque le projet doit implémenter des processus complexes engendrant de multiples imbrications de nombreuses briques. C'est le cas par exemple pour la refonte d'un Back-Office bancaire, la mise en place d'un Front-Office multi-canal, ou encore la construction d'une place de marché.

Cette approche doit également respecter une certaine éthique projet fondée sur les spécificités propres à tout chantier d'envergure : définition des objectifs, organisation pluri-compétences, budget prévisionnel, planning, etc.

Pour réussir un découpage pertinent de ce type de problématique et concevoir des solutions réalistes, il faut relever le challenge de la collaboration entre utilisateurs, fonctionnels et techniques. Les informations échangées doivent être formalisées sans subir la déformation d'un effet « téléphone arabe ».

41 : eXtreme Programming, Rational Unified Process.

Pour éviter ces problèmes de communication liés à des incompatibilités de langage, cette mixité de profils requiert un formalisme commun pendant toute la durée du projet. La création d'un discours universel est une des clés de la capacité à créer une synergie d'équipe. Ce discours devra s'appuyer sur les piliers suivants :

Un formalisme commun comme liant des différents profils du projet.

- Une Méthodologie, cadre général dans lequel les parties progressent de manière itérative et cohérente, chacun l'adressant de son propre point de vue, fonctionnel, applicatif ou technique,

- Un Référentiel des modèles, dans lequel sont répertoriés les modèles formels et la documentation courante,

- Un Vocabulaire, issu des règles communes de représentation.

En pratique, comme nous l'avons évoqué dans un chapitre précédent, nous préconisons le standard UML comme base pour ce formalisme commun. En effet ce langage de modélisation offre toutes les vues nécessaires à la formalisation des volets fonctionnel, applicatif et technique. Ces diagrammes permettent notamment de définir les acteurs (utilisateur, partenaire, fournisseur, etc.), de décrire les processus et les objets métier et de spécifier les flux et les briques applicatives.

C'est avec ce positionnement de pivot central de cohérence que l'architecte s'engage à assumer les responsabilités suivantes :

1. Assurer la compréhension mutuelle avec les acteurs amont, par la normalisation des supports de transmission de l'information,

2. Assurer un rôle de conseil en urbanisme et en Système d'Information pour concevoir des architectures modulaires, faisables au regard des solutions du marché et de l'existant,

3. Assurer un rôle de contrôle et de support dans les phases aval pour garantir la cohésion des réalisations.

La démarche d'architecture projet est classiquement constituée de deux grandes phases majeures. La première, la **phase Etude**, a pour objectif essentiel de cadrer le projet et de définir les plans pour sa mise en œuvre. La deuxième, la **phase Implémentation**, consiste à la réalisation de la solution elle-même.

L'architecture est la pierre angulaire de la phase Etude.

Nous verrons un peu plus loin que c'est sur la phase Etude que la démarche est véritablement la plus innovante. C'est pour cette raison que nous précisons dans le détail chaque activité de cette phase. L'architecte y joue un rôle central de coordination et de communication, assumant les deux premières responsabilités énoncées ci-dessus. Ces activités sont réparties en trois étapes, l'Etude Stratégique, l'Etude d'Architecture Générale et l'Etude d'Architecture Détaillée, telle que le décrit la figure suivante :

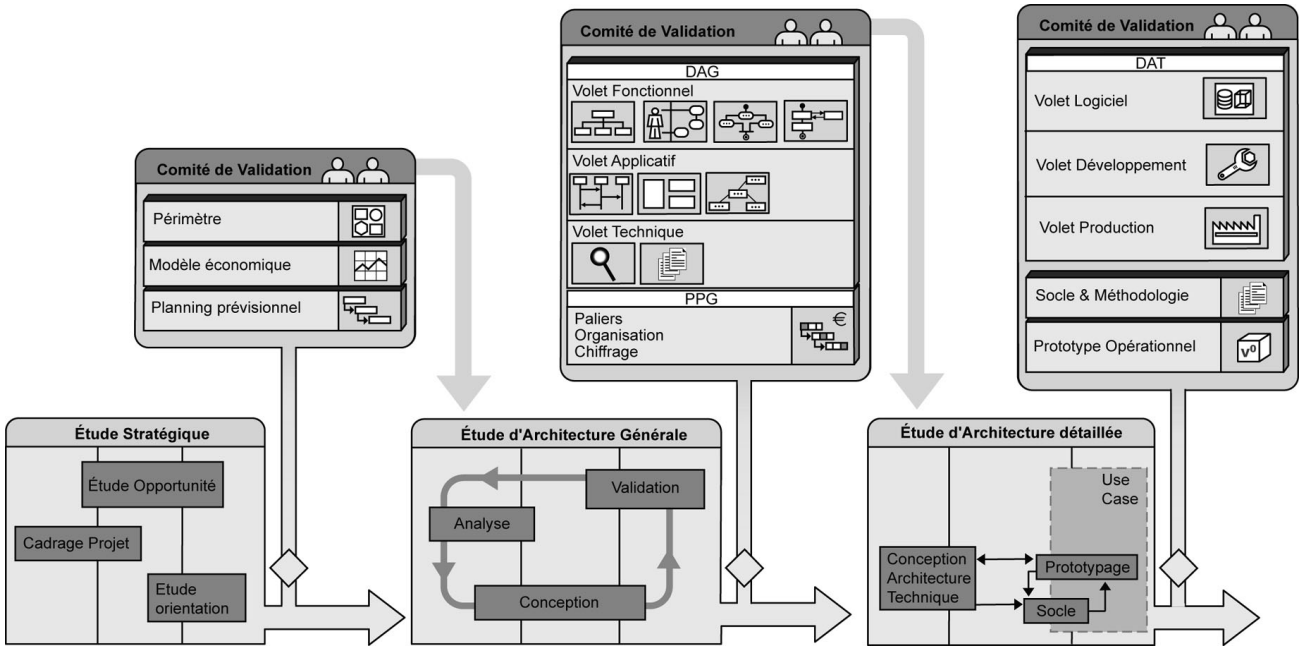


Figure 19 : Les trois étapes de la phase Etude

L'architecture garantit la maîtrise de « l'entropie » dans la phase Implémentation.

Au cours de la phase Implémentation, l'équipe architecture conserve cette responsabilité de conseil en urbanisme et en Système d'Information mais assurera surtout un rôle de contrôle et de support aux équipes de réalisation, garantissant la maîtrise de l'entropie du projet.

L'Etude Stratégique

Pas de projet sans sujet précis.

On ne fait pas de projet sans sujet précis. Lorsqu'un client s'interroge sur l'intérêt de choisir telle solution d'architecture plutôt qu'une autre, il est souvent nécessaire de revenir à des fondamentaux comme : « Pour quoi faire? Pour quand ? Pour qui ? Pour combien ?». Bien que banales, ces interrogations sont rarement soulevées ou alors avec des réponses peu approfondies.

L'étude stratégique : identifier les enjeux, définir les besoins et valider l'opportunité.

Lorsqu'un responsable informatique du milieu industriel souhaite mettre en place un système automatisé de e-Procurement[42], il peut toujours se lancer dans des prototypes et des études pour sentir la maturité du marché. Pour autant, il lui faudra bien un jour répondre à ces questions. A quel type d'achat souhaite-t-il destiner sa future solution ? Quel niveau d'investissement lui semble-t-il raisonnable pour un tel projet ? Quelle rentabilité attend-il de cette évolution ? Quelle est la criticité de mettre en œuvre telle solution ?

Toutes ces interrogations doivent être abordées lors d'une étude stratégique préliminaire, destinée à identifier les enjeux, définir les besoins et valider l'opportunité. Bien que n'ayant aucune vocation à fournir des recettes pour mener cette étude, la démarche d'architecture a besoin de ces résultats pour faire des choix de solutions pertinents. A l'inverse si certains critères ne sont pas pris en compte dès ce préalable au projet, il risque fort de ne pas aboutir ou de passer à côté, malgré tous les efforts que l'on pourrait faire par la suite.

42 : Gestion informatisée des achats.

L'architecte peut participer à cette première étape pour, d'une part, s'assurer que toutes les informations dont il aura besoin par la suite seront bien fournies et, d'autre part, anticiper les contraintes et autres éléments techniques pouvant être structurant pour la suite du projet.

L'étude stratégique fournit des réponses sur les sujets suivants :

1. Le périmètre,
2. Le modèle économique,
3. Le planning prévisionnel.

Cerner le périmètre = rechercher les processus les plus réitérés, à fort impact et nécessitant une traçabilité importante.

Le premier point consiste à définir le périmètre du projet à partir d'une expression de besoin et des objectifs que l'on se fixe pour atteindre ces besoins.

Il est donc nécessaire dans un premier temps de consulter les futurs utilisateurs, ou tout du moins les « fonctionnels », seuls à pouvoir formuler la demande au travers d'une expression de besoin. Il s'agit pour cela de cadrer avec le type de processus que l'on souhaite automatiser, les processus sélectionnés devant présenter plusieurs caractéristiques parmi les suivantes :

- la réitération : le même processus est réalisé plusieurs fois, sans modification ni spécificité,
- la taille : le processus a un impact sur beaucoup de personnes ou d'activités,
- la traçabilité : il est important pour l'entreprise de savoir quelle information a été traitée par qui et quand.

Dans le cadre d'un projet de e-Procurement, l'automatisation de certaines procédures d'achat a plutôt les caractéristiques réitération et taille que traçabilité. Ces deux caractéristiques fortes en font néanmoins un bon candidat à l'informatisation, dans l'espoir d'en tirer un retour sur investissement.

Ce besoin doit ensuite être amendé en vérifiant dans quelle mesure un tel projet pourrait s'inscrire avec cohérence dans la stratégie globale de l'entreprise. Lorsque l'entreprise a investi massivement dans une place de marché, la cohérence d'un projet de e-Procurement est évidente, puisque les gains de productivité attendus le sont principalement du fait de l'homogénéisation des procédures d'achat dans l'entreprise, ce qui prêche pour une certaine centralisation…

Enfin, les contraintes fortes potentiellement structurantes doivent être prises en compte dans cette phase amont. Elles peuvent être d'ordre politique, lorsque l'organisation est impactée, calendaire, lorsque le projet est intimement lié à la roadmap d'autres projets, fonctionnel lorsqu'une partie majeure du processus métier de l'entreprise est en mutation, ou enfin technique lorsque la maturité d'un élément clé du projet n'est pas garantie (par exemple le niveau de sécurité sur Internet).

Cette confrontation des besoins avec la stratégie et les autres dépendances externes permet de définir les objectifs que l'on fixe collégialement pour le projet.

A ce stade de l'étude, il est possible de dresser un premier modèle d'architecture générale, identifiant les principales briques. Le système de GMAO[43], la gestion des fournisseurs, l'intranet Entreprise et la place de marché pourraient par exemple représenter les principaux pavés applicatifs d'un projet de e-Procurement.

43 : Gestion de Maintenance Assistée par Ordinateur.

Construire le modèle économique : un exercice souvent subjectif.

Avec tous ces éléments clairement formalisés, le périmètre du projet peut être précisé, ainsi que les conditions dans lesquelles il est lancé.
Le deuxième thème à approfondir est le modèle économique du projet, ou comment justifier la solution avec le « business case » qu'elle est censée couvrir.

La définition des cas métier adressés permet d'en déduire les processus à mettre en œuvre et donc d'anticiper les réorganisations éventuellement nécessaires. On peut ainsi calculer un investissement prévisionnel et estimer le niveau de ROI à un horizon donné. Comme nous l'avons déjà dit, le calcul du ROI est aisé sur un projet « Tel Que Construit », car fondé sur des métriques connues issues du passé. Il l'est moins sur des projets « Mieux Que Construits » et « Autres Que Construits », pour lesquels des hypothèses doivent être élaborées (réorganisation, diminution du temps de gestion administrative au profit du temps commercial, ouverture de nouveaux marchés, etc.). A ce jeu, on se retrouve souvent dans la peau d'un institut de sondage cherchant son « coefficient correcteur » pour déterminer les intentions réelles de vote... Exercice souvent fallacieux, l'histoire nous l'a montré.

Définir le planning prévisionnel.

Le dernier point consiste à la définition du planning prévisionnel.

A partir d'un premier état des lieux sur la couverture des solutions et des systèmes existants, croisé avec une modélisation du besoin en termes de processus, de criticité et de spécificité, un premier planning d'évolution fonctionnelle est défini, une trajectoire tracée et des grands jalons fixés.

Sensibiliser sur la démarche et le niveau d'engagement de chacun.

Il est indispensable à ce stade de fournir un cadre de travail pour la suite, précisant la démarche, les prérogatives de chaque acteur et leur rôle sur chaque activité prévue dans la démarche. Tous ces éléments doivent être définis et acceptés par tous.

Dans cet ouvrage, nous proposons de nouvelles façons de faire, demandant une forte sensibilisation auprès des intervenants. Dès le départ, il est nécessaire d'anticiper ce besoin d'éducation et de gestion du changement en lançant un réel plan de communication sur la démarche et le suivi de son application.

L'Etude d'Architecture Générale

Les objectifs et les livrables

Trois objectifs pour la phase d'Etude d'Architecture Générale : Concevoir, Communiquer et Chiffrer.

Les objectifs de la phase d'Etude d'Architecture Générale sont de concevoir, communiquer et chiffrer l'architecture générale du projet :

- **Concevoir** la création et la modélisation de l'architecture du système qui servira de guide pour la suite du projet et notamment la phase d'étude d'architecture détaillée,
- **Communiquer** une vision commune de l'architecture générale sur laquelle s'accordent tous les acteurs du projet,
- **Chiffrer** macroscopiquement le projet.

Les résultats de cette phase d'étude sont :

- Le Dossier d'Architecture Générale (DAG),
- La contribution au Plan Projet Général (PPG).

Le Dossier d'Architecture Générale (DAG)

Le DAG (Dossier d'Architecture Générale) : trois angles de vue sur un système unique.

Le DAG se compose de trois volets :

- Un **volet fonctionnel** : il contient des modèles d'analyse fonctionnelle formalisés à partir des expressions de besoin : diagrammes d'organisation, diagrammes de cas d'utilisation, diagrammes d'activité, diagrammes d'états d'objet métier (voir le chapitre Formalisme de modélisation),

- Un **volet applicatif** : il contient des modèles de conception informatique : diagrammes de briques applicatives, diagrammes de séquence, diagrammes de collaboration, (voir le chapitre Formalisme de modélisation),

- Un **volet technique** : il contient des cartographies de l'existant, des audits techniques, des études de re-utilisation d'applications, etc.

Une ou plusieurs architectures ?

Certains ouvrages sur l'architecture de SI introduisent la notion d'architecture fonctionnelle (AF), d'architecture applicative (AA) et d'architecture technique (AT) souvent obtenues en suivant une démarche de construction « top-down »[44].

Ce découpage, bien que séduisant d'un point de vue théorique, nous semble surtout être une vue de l'esprit qui ne reflète pas la réalité de l'architecture de SI où il existe de fortes adhérences entre les différents « niveaux » d'architecture. Aussi, nous préférons considérer l'architecture d'un SI comme un tout indivisible qui peut être observé suivant plusieurs angles de vue ou volet.

Cette distinction peut sembler subtile mais elle est primordiale et va bien au-delà d'une différence lexicale entre le terme « architecture » et le terme « volet ». En effet, non seulement cette approche d'une architecture unique invalide l'existence de ponts de transformation entre les différents « niveaux » d'architecture mais elle a également un impact très fort sur la démarche même de construction de l'architecture. En effet, là où certains conseillent une démarche de construction en cascade AF -> AA -> AT, nous défendons une approche itérative et incrémentale de la construction de l'architecture.

44 : Le projet d'urbanisation du système d'information, de C. Longépé.

Le Plan Projet Général (PPG)

Le PPG est un livrable essentiel qui va contenir et justifier les engagements budgétaires et calendaires du projet, ainsi que son organisation.

Le PPG (Plan Projet Général) : organiser, chiffrer et planifier.

Il contient également trois volets :

- **Volet organisation** : organisation des équipes fonctionnelles, des développements, de l'intégration, de l'homologation. Le volet organisation fait apparaître les éléments transverses, comme le Bureau d'Architecture que nous décrirons plus loin.
- **Volet planning** : lotissement des tâches et plans de migration,
- **Volet financier** : macro-chiffrage du projet,

L'objet n'est pas de présenter ici une méthodologie de chiffrage, mais de résumer comment le Dossier d'Architecture Générale participe à la création de ce document.

Le PPG exploite la structure des blocs de l'architecture, et la description de l'invisible interstitiel, souvent négligé.

Premier aspect, le DAG a défini l'ensemble des briques de l'architecture. Ces briques vont constituer la clé de répartition principale des différents chantiers : par exemple, la brique « référentiel Produit », la brique « infocentre marketing » ou la brique « Sales Force Automation ». Que celles -ci soient implémentés sous la forme de développements spécifiques ou via des progiciels, la **liste des principales fonctions de la brique** permet d'estimer la charge de création, avec l'aide de métriques classiques, dépendantes des méthodes de développement ou d'intégration : Spécifications Fonctionnelles Générales, Spécifications Fonctionnelles Détaillées, Développement/Paramétrage, Tests Unitaires, Tests de briques, Homologation, Migration.
Second aspect, le DAG s'intéresse très en amont aux problèmes d'**intégration entre briques**. C'est souvent à ce niveau que peinent nombre de projets, faute d'avoir anticipé la complexité toujours plus grande de ces chantiers. Plus une brique réalise de tâches , plus l'intégration avec son environnement sera complexe ! L'intérêt du DAG est d'avoir appréhendé la plupart des flux complexes entre briques, et donc de permettre un chiffrage fin pour ces intégrations. Le DAG ne s'intéresse en revanche pas aux flux techniques (réplications de données en particulier), pour lesquels une métrique devra être établie.

Une démarche itérative

La phase d'étude d'architecture générale suit une démarche itérative et incrémentale basée sur les trois activités suivantes :

- **Analyse** des besoins fonctionnels et du SI existant,
- **Conception** de l'architecture générale du SI,
- **Evaluation & Validation** de l'architecture générale.

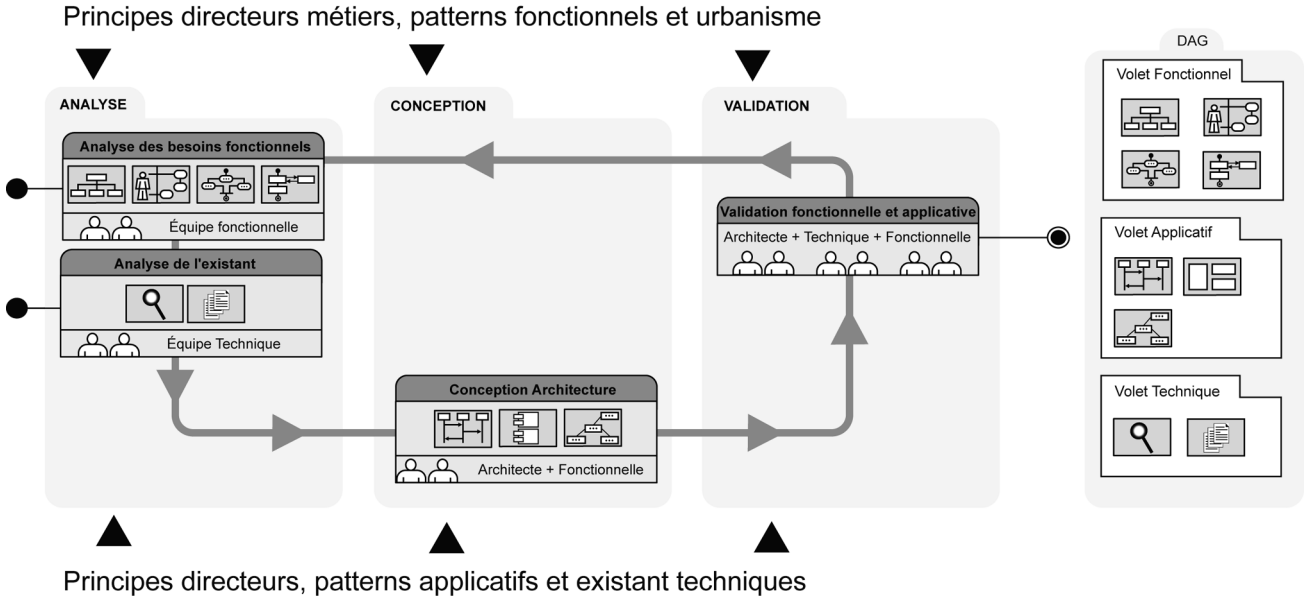


Figure 20 : Démarche de construction de l'architecture générale

Cette approche itérative permet de travailler par zoom successif (macro-processus, cas nominaux, cas d'exception, etc.) et d'affiner au fur et à mesure les modèles de conception de l'architecture générale. Elle favorise également la collaboration entre les différentes équipes participant au projet.

Etant le seul à avoir une vision globale de l'architecture et des contraintes associées (fonctionnelles, applicatives et techniques), c'est l'architecte qui fournit la première ébauche de l'architecture. C'est lui également qui la modifie et l'affine par itérations successives : re-découpage / fusion de briques applicatives, application d'un pattern, mutualisation de flux, etc.
Ces modifications le conduisent à solliciter les équipes fonctionnelles et techniques pour étudier certains aspects particuliers des besoins et de l'existant : demande d'information complémentaire sur un cas d'utilisation, audit technique des applications existantes, étude de flux, etc. Par exemple, l'architecture peut vouloir mutualiser certains traitements communs à plusieurs cas d'utilisation et demandera donc aux équipes fonctionnelles de détailler ces cas d'utilisation à l'aide de diagrammes d'activité pour pouvoir mieux cerner les traitements mutualisables.

Le DAG assure la transition entre la phase d'étude générale et la phase d'implémentation. Il ne se substitue pas au design des briques du système.

Quelle granularité pour l'architecture générale ?

L'Architecture Générale ne décrit pas dans le détail le système mais servira de structure à la phase d'Implémentation.

En particulier, le DAG fait apparaître les différentes briques du SI et leur responsabilité respective. Son rôle n'est pas de rentrer dans le design détaillé de chaque brique. La frontière entre architecture et design s'établit en ayant à l'esprit le principe de subsidiarité : au niveau d'une brique, existe-t-il une volonté transverse désignant l'architecte comme meilleur concepteur plutôt que le futur chef de projet de la brique ?

Pour mieux cerner cette démarche, nous proposons de l'illustrer d'un « use case » qui nous permettra d'attacher des exemples aux concepts évoqués.

Introduction du use case
Paris le 26 septembre 2002, M. Jean-Paul DELEVOYE, ministre de la Fonction publique a adressé un courrier à l'ensemble de la représentation nationale concernant la nécessaire simplification des procédures administratives. Les contraintes, les délais et la complexité de nombreuses procédures sont, en effet, de plus en plus mal acceptés et compris aussi bien par nos concitoyens que par les fonctionnaires eux-mêmes. Le ministre déclare « Simplifions la vie des Français ».

Juin 2003. Malgré l'amélioration sensible des procédures de chacune des administrations concernées (ministère des Finances, caisses d'allocations, ministère de l'Intérieur…) et la mise en place de téléservices de plus en plus nombreux, l'état peine à atteindre l'objectif fixé, car aucune de ces initiatives n'est coordonnée. Un changement d'adresse ou de situation familiale reste toujours un cauchemar pour l'usager. Le guichet unique de la Fonction Publique n'existe pas.

Septembre 2003. Après un débat houleux à l'assemblée, la question du guichet unique a trouvé son compromis. Aucune administration ne pouvant légitimement se démarquer des autres pour gérer une relation usager et s'interfacer aux autres acteurs, cette mission a été confiée à trois piliers de la vie quotidienne des citoyens : les mairies, les préfectures, et les bureaux de Poste. Chacun aura le même niveau de prérogatives, et distribuera donc les mêmes prestations. Le service sera gratuit pour l'usager, et financé par un transfert budgétaire des administrations vers les guichets, au titre des économies liées à la baisse de fréquentation directe. Le guichet fournit à l'usager les prestations transverses et un accueil de premier niveau, les administrations conservant la mainmise sur leurs spécificités : recherche d'emploi, déclaration fiscale, scolarisation, justice, etc.

Janvier 2004. Le projet GNU « Guichet National Unifié » sera réalisé à l'extérieur des Systèmes d'Informations des entités concernées pour mieux être partagé. Les opérateurs des mairies, des préfectures et de La Poste utiliseront ce système unique, simplifiant ainsi la mobilité du citoyen, tout en réduisant les coûts de mise en place. Dans un premier temps, les guichets proposeront aux usagers la prise en charge de l'ensemble de leurs formalités, mais n'en automatiseront qu'une partie.

Le système d'information qui sous-tend l'activité aura un premier palier opérationnel à un horizon de deux ans. Un second palier renforcera l'aspect relation au client avec la mise en place d'un système proactif (« push »).

La phase d'analyse

La phase d'analyse mène en parallèle l'analyse fonctionnelle et l'analyse technique de l'existant.

La phase d'analyse mène deux activités en parallèle :

- Recueil et analyse des besoins fonctionnels,
- Audit et analyse de l'existant applicatif et technique.

L'objet n'est pas de présenter une méthodologie de recueil de besoins fonctionnels mais de montrer l'information nécessaire à l'architecte pour concevoir l'architecture du SI. En effet, tous les besoins exprimés par les équipes fonctionnelles n'ont pas nécessairement d'impact sur l'architecture générale. Par exemple, spécifier que le nouveau système d'information d'une entreprise doit pouvoir manipuler aussi bien des euros que des dollars pour la gestion des commandes, ne doit donc pas être pris en compte pendant la phase de conception, car il ne s'agit pas d'un besoin structurant à l'échelle du SI.

L'architecte a donc un rôle de sélection et de structuration des besoins fonctionnels afin de faire ressortir ce qui contraint l'architecture du système. Cette activité est menée en étroite collaboration avec les équipes fonctionnelles afin de garantir la validité des modèles fonctionnels utilisés.

L'architecte est en particulier intéressé par :

- **L'organisation de l'entreprise** : il ne s'agit pas de décrire dans le détail l'organigramme de l'entreprise mais plutôt de donner une vue d'ensemble du découpage organisationnel de l'entreprise sur le périmètre du projet. L'architecte s'intéresse également aux différents acteurs qui seront amenés à utiliser le système d'information.

L'architecture ne s'intéresse pas à tout les cas fonctionnels.

- **Les cas d'utilisation** : le recueil des besoins fonctionnels s'organisant généralement autour des cas d'utilisation (voir le chapitre « Formalisme de modélisation »), l'architecte cherche à les trier, les sélectionner et regrouper ensemble ceux qui ont une forte affinité fonctionnelle. Tous les cas d'utilisation ne sont pas forcément pertinents pour construire l'architecture générale. Il suffit d'étudier les principaux cas d'utilisation métier. Il est par contre important de les choisir avec soin pour qu'ils soient représentatifs de l'ensemble du métier : 2 ou 3 cas d'utilisation par processus métier, par exemple.

- **Les objets métier** : les objets métier auxquels l'architecte s'intéresse sont ceux qui ont un sens pour les équipes fonctionnelles. Il ne s'agit pas de « deviner » les objets internes du système mais de se concentrer sur ceux qui apparaissent directement dans le discours des fonctionnels. Ces objets métier se retrouveront au cœur de la problématique d'interopérabilité entre briques.

En parallèle de la phase d'analyse fonctionnelle, les architectures pilotent des audits de l'existant applicatif et technique. Ces études sont d'autant plus importantes que le poids de l'existant est lourd. Dans le cas de systèmes « from scratch », les études porteront plutôt sur les SI avec lesquels ils doivent s'interfacer : services offerts, formats d'échanges, référentiels existants, etc.

Illustration de la phase d'analyse sur le use-case GNU

Recueil des besoins fonctionnels

Parmi les documents fournis par les équipes fonctionnelles, l'architecte va s'intéresser particulièrement à ceux décrivant l'organisation fonctionnelle, les cas d'utilisation et les différents acteurs.

L'administration GNU (Guichet National Unifié) est composée de trois directions :

- **Direction des services aux usagers :** c'est la direction chargée des relations avec les usagers et des services qui leur sont rendus.
- **Direction des opérations :** c'est la direction chargée de l'organisation des guichets, des utilisateurs du SI et des infrastructures.
- **Direction des relations avec les administrations :** c'est la direction chargée des relations avec les correspondants des administrations interfacées à GNU (gestion, refacturation et traitement des procédures).

Les principaux utilisateurs du système GNU sont :

- **Usager :** citoyen bénéficiaire des prestations administratives.
- **Utilisateur :** guichetier qui sert les usagers.
- **Administrateur :** guichetier aux droits plus élevés qui peut utiliser des fonctions d'administration.
- **Correspondant administratif :** interface entre GNU et les administrations publiques.

Voici des exemples des principaux cas d'utilisation du système, fournis par les équipes fonctionnelles :

- **Ouverture de dossier usager :** création d'un dossier usager dans le système GNU.
- **Suivi de dossier usager :** modification, consultation et fermeture d'un dossier usager.
- **Demande de prestation :** ouverture d'une demande de prestation offerte par GNU. Par ex. : changement de statut familial, changement d'adresse, demande de duplicata, etc.
- **Suivi de prestation :** modification et consultation des prestations en cours.
  Création d'une nouvelle prestation : mise à disposition d'une nouvelle prestation administrative.
- **Identification & habilitation :** authentification et identification des usagers, des utilisateurs, des administrateurs, etc. Chaque personne possède des habilitations propres.
- **Refacturation de procédure :** les prestations administratives exécutées par GNU sont refacturées aux administrations centrales.

Chaque cas d'utilisation est généralement accompagné d'une description textuelle voire d'un diagramme d'activités qui montre l'enchaînement des activités participant à la réalisation du cas d'utilisation.

Analyse des besoins fonctionnels

Comme indiqué dans la démarche, la première tâche de l'architecte est de trier, structurer et analyser les expressions de besoins fournies par les équipes fonctionnelles.

Parmi les acteurs décrits par les équipes fonctionnelles, l'architecte retient ceux qui sont acteurs principaux du système au sens UML, c'est-à-dire ceux qui interagissent directement avec le système. Dans le cas du système GNU, il s'agit donc de l'utilisateur, de l'administrateur et des correspondants administratifs. Par contre, l'usager final n'interagissant pas directement avec le système, il ne sera pas considéré pendant cette phase d'étude générale[45].

L'étude des cas d'utilisation permet de faire ressortir les principaux objets métier du système GNU :

- **Dossier usager :** contient les informations relatives à l'usager, l'historique des prestations, ainsi que la liste des prestations en cours de traitement.
- **Prestation :** une prestation désigne le regroupement de plusieurs procédures administratives. Exemple : la prestation « Déclaration de naissance » contient les procédures « Déclaration de naissance pour l'état-civil » et « Déclaration de naissance pour la CAF », etc..
- **Procédure :** une procédure désigne un service mis à la disposition de GNU par une administration publique. Exemple : « Changement de statut familial pour l'état-civil» ou « Demande de duplicata de permis de conduire ».
- **Note :** nous avons écarté l'acteur « Usager » pour notre étude générale mais nous conservons l'objet métier « Dossier usager ».

La procédure administrative est centrale pour l'activité du GNU, et constitue donc un objet métier dont le cycle de vie est décrit ci-après :

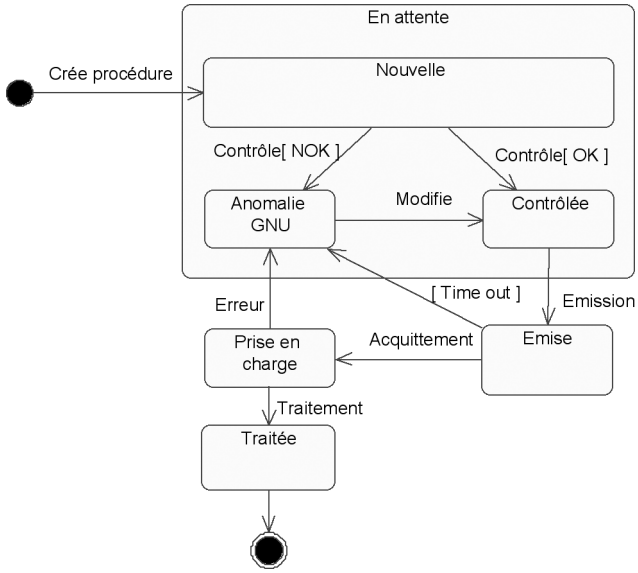


Figure 21 : Diagramme d'état pour l'objet métier « Procédure »

45 : On s'y intéressera plus tard à un niveau technique pour matérialiser les accès libre services (Internet et bornes interactives).

Ensuite, l'architecte trie et organise les cas d'utilisation en regroupant ceux liés par une affinité fonctionnelle forte (manipulation du même objet métier, même contexte fonctionnel, etc.) :

Package « Gestion des usagers »

- Ouverture de dossier usager
- Suivi de dossier usager

Package « Gestion des prestations »

- Demande de prestation
- Suivi de prestation
- Changement de statut familial, Changement d'adresse, Demande de duplicata, etc.
- Création d'une nouvelle prestation

Package « Sécurité »

- Authentification usager
- Authentification utilisateur
- Authentification administrateur

Package « Gestion des procédures »

- Mise en place d'une nouvelle procédure
- Traitement d'une procédure
- Refacturation de procédure

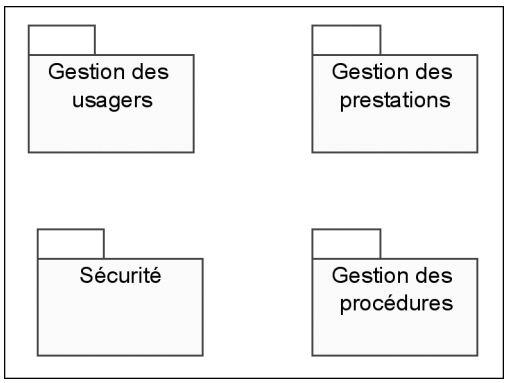


Figure 22 : Regroupement en packages des cas d'utilisation

Chaque package fait l'objet d'un ou de plusieurs diagrammes qui décrivent les cas d'utilisation regroupés ainsi que les relations qu'ils ont entre eux :

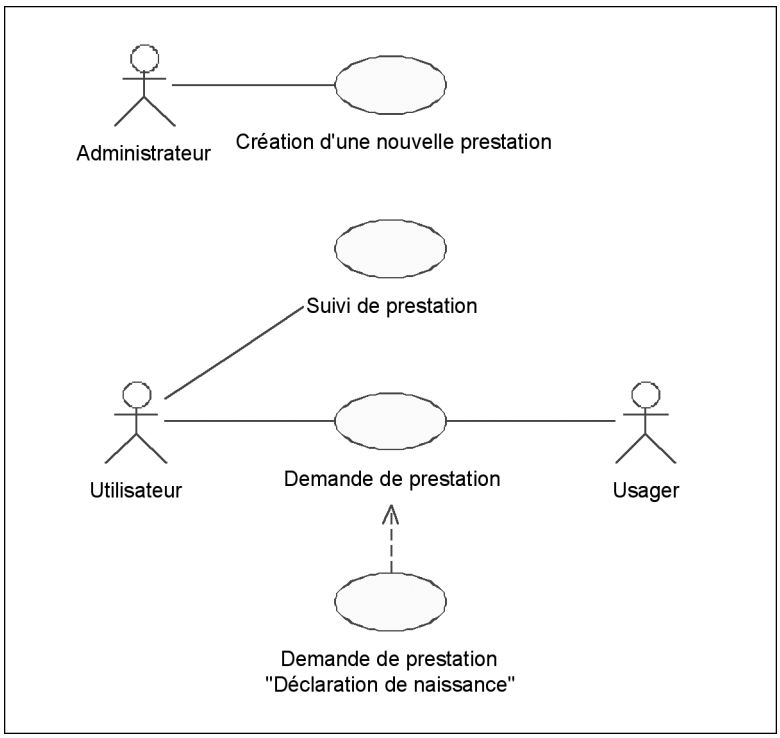


Figure 23 : Diagramme de cas d'utilisation pour le package « Gestion des prestations »

La phase de conception

La phase de conception est initiée par l'architecte à partir des modèles de la phase d'analyse.

Une première réflexion fondée sur les différents modèles d'analyse fonctionnelle (organisation de l'entreprise, principaux objets métier, regroupement des cas d'utilisation en packages, etc.) permet à l'architecte d'ébaucher un premier découpage du SI en briques applicatives.

Ce découpage est nourri par les modèles de la phase d'analyse fonctionnelle mais, comme dans tout travail de création, il est également le résultat de considérations moins rationnelles qui influencent l'architecte : expérience, intuition et orientation du projet.

Cette première version de l'architecture permet d'initier le cycle d'itération qui va permettre d'affiner et d'améliorer l'architecture générale du SI.

La démarche itérative de conception d'architecture se base sur la construction de modèles complémentaires. Chaque nouveau modèle permet de changer de point de vue sur l'architecture et de détecter des erreurs ou des opportunités d'amélioration :

- **Les diagrammes de briques applicatives** permettent de décrire l'architecture statique du système.
- **Les diagrammes de séquence** qui donnent une vision dynamique du système, vont par exemple faire ressortir les opportunités de mutualisation de traitement.

- **Les diagrammes de collaboration** permettent de modéliser le contexte des briques applicatives et en particulier de contrôler/maîtriser les flux qu'ils échangent avec le reste du système.

Des itérations successives avec les différents acteurs du projet permettent d'affiner l'architecture générale du SI.

Chaque modification ou amélioration de l'architecture peut conduire l'architecte à demander des études complémentaires aux équipes fonctionnelles et techniques. C'est cet aspect itératif qui fait la force de cette démarche.

Rappelons qu'en termes de granularité, l'architecture générale s'arrête au niveau de la brique applicative, c'est-à-dire une entité cohérente fonctionnellement et techniquement. Elle pourra par la suite faire l'objet d'un choix de progiciel ou d'un développement spécifique, nécessitant dans ce cas une spécification plus détaillée.

Illustration de la phase de conception sur le use-case GNU

Première ébauche d'architecture générale

Une première réflexion fondée sur les différents modèles d'analyse fonctionnelle (organisation de l'entreprise, principaux objets métier, regroupement des cas d'utilisations en packages, etc.) permet à l'architecte d'ébaucher un premier découpage du SI en briques applicatives :

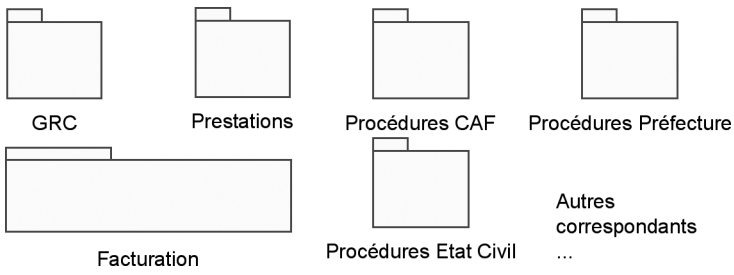


Figure 24 : Diagramme de briques applicatives

Il s'agit ensuite d'éprouver et de vérifier ce découpage en considérant maintenant le système dans un contexte dynamique. Pour cela, l'architecte construit les diagrammes de séquence associés aux principaux cas d'utilisation.

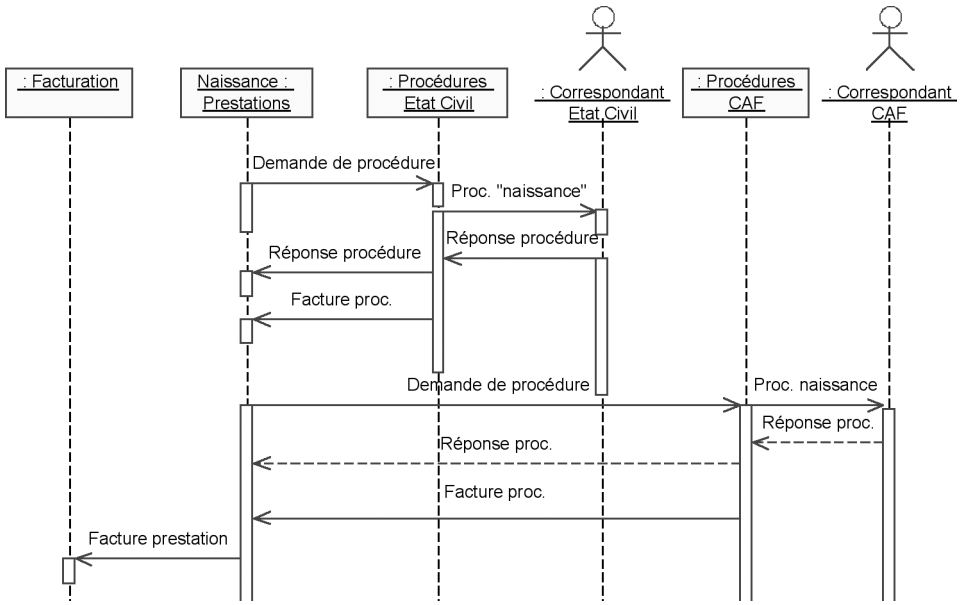


Figure 25 : Diagramme de séquence pour le cas d'utilisation « Déclaration de naissance »

Première itération d'architecture générale

On remarque sur le diagramme de séquence que le traitement de la procédure « naissance pour l'état civil » est très similaire au traitement de la procédure « naissance pour la CAF ». Cette observation conduit l'architecte à se demander comment cette architecture pourrait être améliorée. Dans un but de mutualisation des traitements, l'architecte demande aux équipes fonctionnelles de préciser le traitement des procédures « naissance pour l'état civil » et « naissance pour la CAF » à l'aide de diagrammes d'activité.

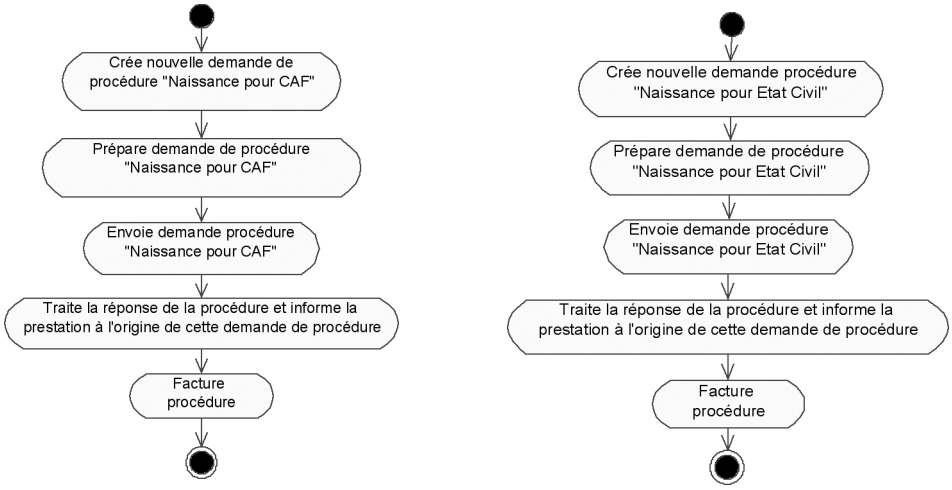


Figure 26 : L'observation de ces diagrammes d'activité confirme cette similitude.

Sous la direction de l'architecte, une analyse des relations entre le cœur du système (« GRC » « Prestations ») et les briques applicatives qui permettent de communiquer avec les administrations distantes, fait ressortir les points suivants :

- Chaque administration a son propre mode d'accès (télé-procédure, email, courrier, etc.) et format d'échange (la CAF utilise le numéro de sécurité sociale pour identifier l'usager alors que le centre des impôts possède un identifiant contribuable), cependant toutes offrent une notion transverse de procédure.

- Il y a donc opportunité de mutualiser les traitements communs à toutes les administrations distantes : création et suivi de la procédure, refacturation de la procédure, etc.

- Il existe de fortes contraintes de sécurité : le GNU doit être un acteur de confiance à qui les administrations délèguent une partie de leurs procédures, le GNU doit donc créer une zone de confiance virtuelle entre les différents Systèmes d'Information.

L'architecte suggère l'utilisation d'un pattern « Royaume / Emissaire » qui permet de découpler la gestion des prestations usagers, du traitement spécifique des procédures par les administrations distantes. L'architecte propose donc une nouvelle version de l'architecture après application du pattern :

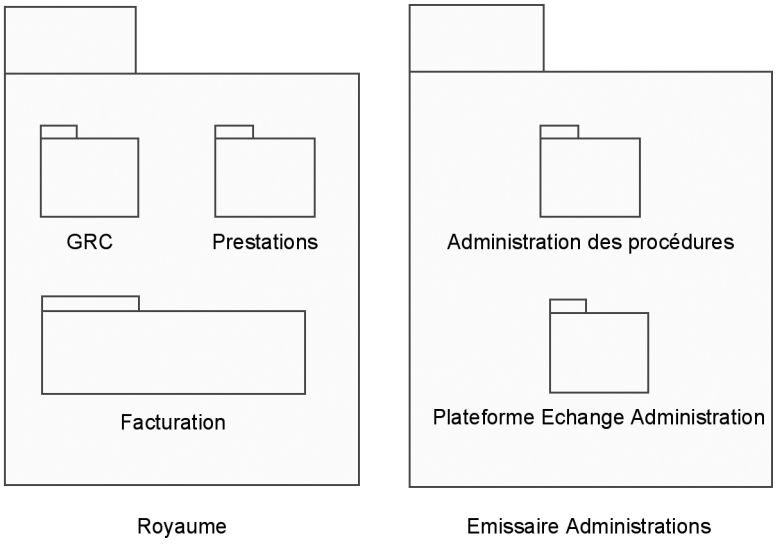


Figure 27 : Application du pattern Royaume/Emissaire

A ce stade, le système peut supporter une organisation distribuée dans les guichets pour les fonctions du Royaume, et une organisation distribuée ou centralisée pour l'Emissaire.

Architecture générale finalisée

Plusieurs itérations permettront ensuite d'affiner ce modèle statique pour aboutir au modèle final :

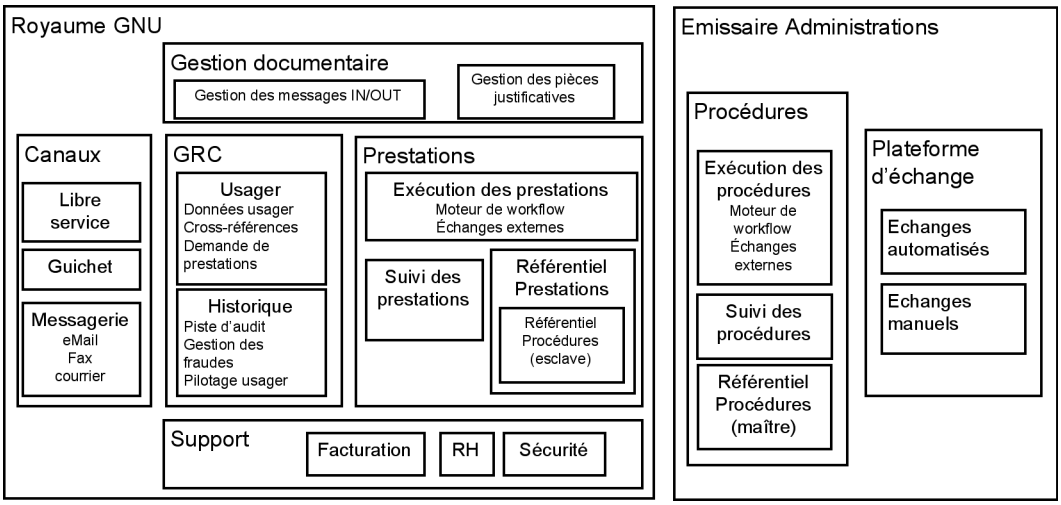


Figure 28 : Diagramme final de briques applicatives

Chaque brique fait également l'objet d'une fiche détaillée, mentionnant notamment les principaux services rendus, un commentaire sur les points clés, etc.

Enfin, nous identifierons à ce stade les **éléments clés de l'architecture technique** : ce sont les éléments « sensibles » pour lesquels des choix structurants doivent être pris en début de projet car susceptibles d'impacter les chiffrages. Ils seront validés par le prototype.
En ce qui concerne le guichet GNU, l'architecte aura identifié les points clés suivants :

- **La gestion du multi-entités**
  Le fait de pouvoir initier une prestation dans une entité (Préfecture) et de pouvoir suivre cette prestation dans une autre entité (Mairie), ou sur le Web, conduit inévitablement à une architecture centralisée. Cette centralisation devra néanmoins autoriser un cloisonnement logique des entités au sein du système unique. Le multi-entité promet des gains par rapport à des systèmes autonomes, mais il alourdit notoirement la problématique d'habilitation.

- **L'achat ou le développement**
  Certains composants d'infrastructure clé en main seront clairement intégrés dans l'architecture (worflow, échanges, base de données, GED...), qu'ils soient issus d'éditeurs ou du monde Open Source. Quant aux fonctions applicatives, au vu des spécificités du traitement des prestations et des procédures, seule la partie GRC pourra exploiter les fonctions d'un progiciel métier. Le choix se portera sur un acteur dédié à la gestion de base client et d'historique, plutôt qu'un acteur généraliste.

- **La gestion optimisée du multi-canal**
  Dans un contexte ou l'accès aux services peut se faire soit par des canaux « Libre Service » (Web, Minitel, bornes interactives, etc) soit par des canaux « Guichet » ou centre d'appel, il est essentiel de pouvoir factoriser au maximum le SI et d'éviter les développements redondants dans chaque filière.

- **La gestion des échanges avec les partenaires**
  La plate-forme d'échanges est le point central de communication avec les Administrations et autres partenaires (notaires, RSS, etc.). Il s'agit ici de déterminer les formats d'échanges utilisés (syntaxe, sémantique) et la sécurisation des échanges (authentification, confidentialité, intégrité, non-répudiation...).

- **La gestion documentaire**
  La numérisation et la gestion des documents est également transverse au système : gestion des messages entrants/sortants avec les usagers et les partenaires (formulaires, pièces justificatives…).

- **La gestion du Workflow**
  Modélisation, exécution et pilotage des processus pour l'exécution des prestations et des procédures, suivi des états des différentes instances de processus, gestion des tâches humaines, processus d'escalade, etc.

La phase de validation

L'architecture générale pose les fondements d'un système d'information. C'est en effet à ce niveau que sont prises des décisions structurantes pour le système : utilisation de patterns, mutualisation de briques fonctionnelles ou d'infrastructure, définition des mécanismes d'interopérabilité inter-modules, etc. Dans le but de maîtriser les risques projet, le directeur de projet est en droit d'obtenir une certaine garantie quant à l'**adéquation** et la **qualité de l'architecture générale** proposée.

L'architecture générale doit être évaluée et validée selon deux axes : un axe fonctionnel et un axe Système d'Information.

Ceci conduit donc l'architecte à devoir répondre aux deux questions suivantes :

- **Mon architecture répond-elle aux besoins fonctionnels du système pour laquelle elle a été construite ?** Couverture fonctionnelle et respect de principes fonctionnels structurants (sécabilité, multilinguisme, multi-entité, etc.)

- **Mon architecture répond-elle aux objectifs de qualité du SI pour laquelle elle a été construite ?** Critères de flexibilité, critères de sécurité, standards technologiques, performance, accessibilité, etc.).

La validation de l'architecture doit demeurer une activité continue, même si une phase formelle peut ponctuer la conception.

Qui valide le DAG ?

La validation du DAG est sous la responsabilité du directeur de projet en s'appuyant à la fois sur les équipes fonctionnelles et techniques. Toutefois, ces dernières n'en valident pas les mêmes aspects. La validation du DAG peut s'effectuer de manière formelle à la fin de la conception, mais il est pertinent de dérouler cette activité tout au long du projet. C'est le principe même d'une démarche itérative.

Comment valider un DAG ?

Il existe de nombreuses façons de valider une architecture technique (prototype, benchmark de performance, etc.) mais il est plus difficile de trouver des méthodes pour valider une architecture sous tous ses volets.

Nous proposons une méthodologie de validation à deux niveaux pour répondre aux deux questions qui ont été précédemment posées :

- **La validation suivant un axe fonctionnel** permet de vérifier que l'architecture considérée répond bien aux besoins fonctionnels,
- **La validation suivant un axe SI** permet de vérifier que l'architecture répond aux objectifs de qualité du SI.

Validation suivant un axe fonctionnel

Le diagramme de séquence est au centre de la démarche de validation.

Qui :

Equipe fonctionnelle et architecture

Comment :

Les besoins fonctionnels impactants étant exprimés par des cas d'utilisation, il s'agit donc de construire les diagrammes de séquence illustrant ces use-cases. Bien sûr, il ne s'agit pas de considérer tous les cas d'utilisation. Les principaux cas nominaux (règle des 80 / 20) et quelques cas d'exception suffisent à valider l'architecture générale d'un point de vue fonctionnel.

Cette opération peut être réalisée sous la forme d'ateliers entre les équipes fonctionnelles et techniques, où l'on éprouve collégialement les cas d'utilisation sur l'architecture. **Le diagramme de séquence est au centre de la démarche de validation**.

Validation suivant un axe SI

La caractérisation de métriques techniques permet de valider l'architecture d'un point de vue SI.

Qui :

Equipe architecture + équipe technique

Comment :

Une architecture générale peut couvrir les besoins fonctionnels sans être optimale d'un point de vue Système d'Information. Il est donc nécessaire d'évaluer l'architecture suivant cet axe pour arbitrer entre plusieurs conceptions et/ou qualifier une architecture par rapport à des critères définis dans le cahier des charges. Par exemple, tirer sur l'axe flexibilité va nécessairement impacter l'axe coût, on a rien sans rien.

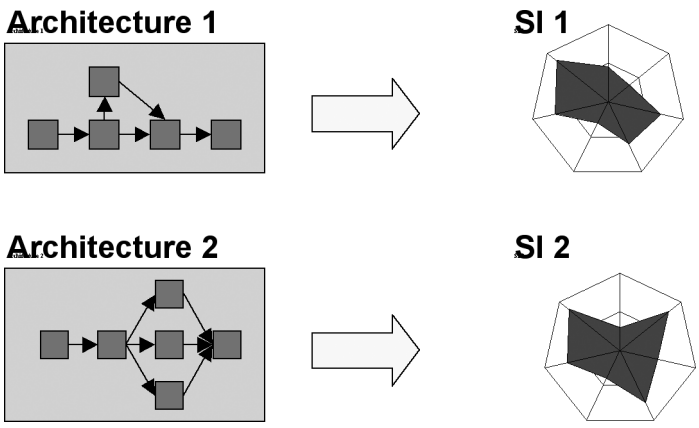


Figure 29 : Evaluation d'architectures par caractérisation

Il existe de nombreuses façons de caractériser un SI. Voici une liste des métriques qui nous semblent pertinentes mais non exhaustive dans le cadre de l'architecture générale :

- **Agilité / Extensibilité :**
  - Capacité à intégrer rapidement de nouveaux flux et/ou de nouveaux applicatifs : intégration d'un nouveau canal de distribution, sous-traitance d'une activité, etc.
  - Capacité à supporter plusieurs modes d'organisations,
- **Respect des standards applicatifs et techniques de l'entreprise**
  - L'architecture utilise-t-elle les fonctions transverses mises en place au niveau SI, comme les référentiels, la sécurité, les plates-formes d'échanges, les services communs ?
  - L'architecture technique s'insère-t-elle dans l'univers d'exploitation connu ou nécessite-t-elle de nouvelles technologies ?
- **Evolutivité :** Capacité à intégrer de nouveaux utilisateurs, des référentiels plus larges, montée en charge, etc.
- **Sécurité :** Mesure de la sécurité globale du système : sécurisation des échanges, difficulté d'intrusion, authentification des acteurs, etc.
- **Coûts** : Mesure du rapport qualité/prix, benchmark par rapport à d'autres constructions.

Enfin dans le cas d'une architecture transitoire, il est également nécessaire de considérer la métrique « **Ecart par rapport à la trajectoire** » : des briques non pérennes sont-elles apparues ? A-t-on avancé vers la cible ?

Remarques sur la formulation des objectifs de qualité du SI

Il est important de noter que ces métriques n'ont de sens que par rapport à des critères d'évaluation précis. Ceux-ci sont exprimés dans le cahier des charges du système mais souvent de manière imprécise. Par exemple : « Le SI devra être construit sur un socle évolutif et pérenne aussi bien d'un point de vue fonctionnel, applicatif et technique ». .

Exprimés de cette façon, les objectifs du SI sont flous et peu directeurs. En effet, qui voudrait d'un système d'information construit sur un socle dépassé sans possibilité d'évolutions ?

De la même façon, exiger d'une architecture qu'elle soit sécurisée est trop imprécis. Une architecture n'est dite sécurisée que vis-à-vis de certains risques. Par exemple, une architecture peut s'appuyer sur un excellent système d'authentification et d'habilitation mais rester vulnérable à une attaque interne.

Il est donc indispensable pour évaluer – et donc valider – un système d'information de disposer d'objectifs et de critères de qualité précis. Ceux-ci sont en général exprimés dans le cahier des charges. Par exemple, au lieu de dire « le SI devra être modulaire », il est préférable de dire « le SI devra pouvoir permettre l'externalisation de la gestion des expéditions sous 5 ans ». De même, au lieu de dire « le SI devra être scalable », il est préférable de dire « le SI devra être capable de permettre une augmentation de 200 % du nombre de transactions dans les 3 ans ».

L'Etude d'Architecture Détaillée

L'Etude d'Architecture Détaillée est la dernière partie de la phase Etude, elle se fonde notamment sur le Dossier d'Architecture Générale, et va s'attacher à :

- Concevoir l'architecture technique du SI sur la base du DAG,
- Mettre en place un socle technique et méthodologique opérationnel pour la phase Implémentation,
- Valider cette architecture et ce socle (technique, méthodologique) sur la base d'un prototype implémentant un use-case significatif.

Les livrables de cette phase sont constitués d'une partie documentaire et d'une partie opérationnelle :

- Le **Dossier d'Architecture Technique** (DAT), organisé en trois volets (Logiciel, Développement, Production) qui décrivent respectivement les mécanismes logiciels de l'architecture technique (chaînes de liaison, produits utilisés), le framework de développement et son cadre d'utilisation (outils, librairies, méthodologies), et les caractéristiques de l'architecture de production (aspects systèmes et réseau, dimensionnement, implantation des composants, supervision du système),

Le prototype sert de support à la validation de l'architecture, et de la méthodologie.

- Une version opérationnelle du **socle technique et méthodologique** : environnement de développement, de tests et d'intégration, frameworks et services techniques communs (IHM, logs…), services « métier » communs (parseur de messages, machine à états…), guides méthodologiques,
- Un prototype opérationnel implémentant **un use-case significatif**, permettant de valider les choix d'architecture définis dans le DAT, les exigences de production (qualité de service, montée en charge, déploiement…), d'éprouver le socle technique (outils, frameworks…) sur une réalisation concrète, de valider la méthodologie sur l'ensemble de la chaîne de production (des spécifications techniques aux procédures de déploiement).

`

Le framework de développement, un élément fondamental pour la réussite du projet
Un framework correspond à un ensemble d'outils du marché, de bibliothèques spécifiques et de méthodologies qui visent à faciliter, cadrer et accélérer les développements du projet.
Le framework de développement, élaboré en phase Etude et consolidé en phase Implémentation, est fondamental à la bonne réussite du projet :

Il définit le cadre de travail et les engagements de chacune des parties, fonctionnelles et techniques, en spécifiant le processus de développement.
Le processus de développement décrit le « qui fait quoi ? » : quelle est la profondeur des Spécifications Fonctionnelles Générales ? Comment sont-elles formalisées (Word, UML) ? Produit-on des Spécifications Fonctionnelles Détaillées ou se suffit-on d'une documentation générée à partir du code ?

Il permet de mieux décorréler les aspects techniques de l'implémentation des fonctionnalités.
Un framework a pour objectif de fournir un ensemble de services, de factoriser les difficultés, et par conséquent d'affranchir les concepteurs et les développeurs de la résolution de problématiques techniques ou récurrentes. Au niveau de l'exploitation des compétences, cette approche permet d'utiliser des développeurs moins expérimentés, dans la mesure où chacun n'a pas à connaître la complexité sous jacente des frameworks utilisés.

Il contraint et standardise les développements.
Le cadre d'utilisation du framework doit permettre de canaliser le projet et d'éviter la dispersion des équipes. La présence de ce cadre augmente la réutilisation, et évite le cloisonnement entre les différentes équipes de développement

Il diminue les risques liés au projet.
L'utilisation de différents frameworks reconnus est un gage de qualité. En effet, tout bon framework (issu d'organismes de normalisation métier, de l'Open Source, ou le fruit d'un éditeur) aura été l'objet de longs efforts et de compromis d'experts. Même si la mise en place d'un framework requiert un ticket d'entrée initial pour les projets, il se rentabilise toujours par une meilleure maîtrise des délais et de la qualité.

Le Dossier d'Architecture Technique

Ce dossier sera composé de trois volets (Logiciel, Développement, Production), chaque volet étant guidé par un fil conducteur : la structuration en couches de l'architecture technique.

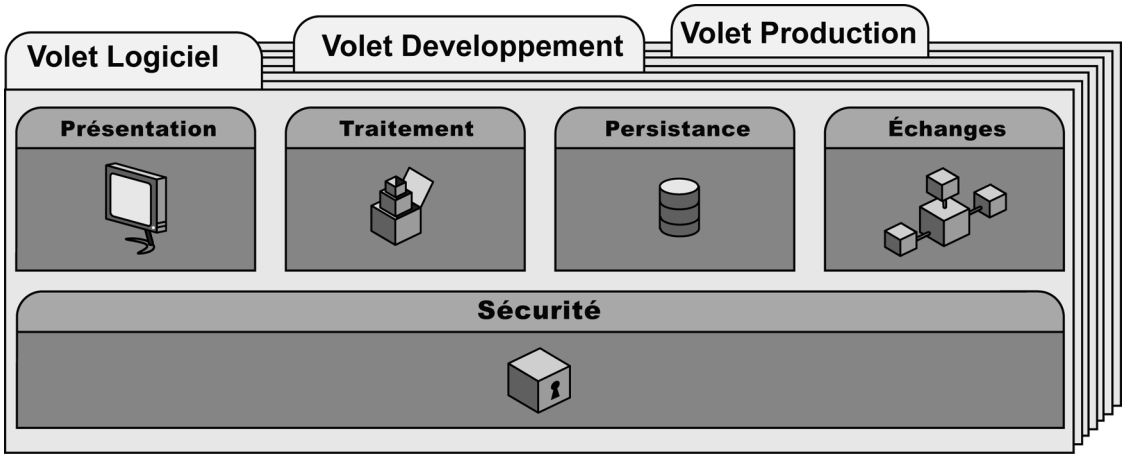


Figure 30 : Structure d'un Dossier d'Architecture Technique

Dans ce modèle que nous proposons, chaque couche implémente des services spécifiques, dont nous donnons une liste non exhaustive. Le rôle de l'architecte étant précisément de les définir en détail lors de la conception de l'architecture.

Un dossier structuré en 5 couches, les volets développement et production permettent une lecture transverse.

- **Couche Présentation**
  Services IHM (client lourd, client léger), gestion du multi-canal (web, voix, mobile, fax), services de portail (aggrégation d'IHM, bouquets de services), services de reporting, services d'impression (impressions PDF, gestion de templates, éditions de masse…),
- **Couche Traitements**
  Implémentation de la logique métier sous forme de composants stateless/statefull, pooling des composants, exposition des composants (Web Services, RMI/IIOP, .NET Remoting…) à d'autres applications (couche Présentation, couche Traitements), services d'accès aux données (encapsulation des requêtes SQL, voire mapping Objet/Relationnel…), services transactionnels (transactionnel SGBD, transactionnel multi-ressources en série ou en parallèle[46]),
- **Couche Persistance**
  Services de stockage des données, moteurs relationnels, bases objets, bases XML…
- **Couche Echanges**
  Services d'échange entre Traitements (échanges synchrones, asynchrones), entre système de Persistance (synchronisation de référentiels, ETL...), services de garantie de livraison de message, Message Broker (Transformation, Routage, DataFlow), services de gestion de transactions étendues (processus, compensation),
- **Couche Sécurité**
  Services de sécurité : SSO, authentification, gestion des habilitations, intégrité, non-répudiation… La sécurité n'est pas une couche isolée, mais transverse aux autres couches : authentification des utilisateurs et contrôle des habilitations au niveau des services IHM, sécurisation des traitements (authentification, habilitations grosse maille et habilitations fines…), sécurisation des échanges, sécurisation des données…

Le volet Logiciel

Le volet Logiciel décrit les logiciels utilisés (serveurs d'applications, EAI, SGBD, etc.), et les chaînes de liaison entre les éléments de l'architecture (interactions entre couches), ainsi qu'avec le reste du SI.
Nous ne fournissons pas ici de modèle formel (UML...) pour ce volet ; hormis la structuration en couches, décrivant les mécanismes logiciels pour les différents services d'infrastructure identifiés. A titre d'exemple, on pourra trouver au niveau de la couche Présentation la description technique des services d'impression (trop souvent considérés à tort comme la dernière roue du carrosse !) : librairies StreamServe pour la génération de documents PDF à partir de modèles, librairies Crystal Report pour les services de reporting, description des chaînes de liaison entre les différents éléments de l'architecture, description du flux extranet avec un partenaire pour l'édition de masse, etc.

46 : On parle alors de two-phase commit, protocole lourd à n'utiliser qu'en dernier recours.

Exemples de points clés du Volet Logiciel pour le use-case GNU

Gestion du multi-entités (couche traitement)
L'architecte proposera la mise en place d'un annuaire LDAP centralisé contenant les utilisateurs, leurs rôles et leur entité. Le système étant bâti sur des briques récentes, cet annuaire pourra être unique, évitant l'usage d'un progiciel de « provisioning » pour propager les informations utilisateurs dans différentes cibles (GRC, GED, workflow..). Les librairies d'authentification et de gestion des habilitations (récupération des droits d'accès, gestion des modèles d'habilitations spécifiques à chaque entité, etc.) seront incluses dans le socle technique et utilisées par l'ensemble des applications. Le choix d'une organisation centralisée pour les correspondants administration n'impose pas le support du multi-entité dans l'Emissaire.

Gestion mutualisée du multi-canal (couche IHM)
Une couche spécifique de « services IHM » exposera sous forme de services Web des fonctions dédiées « enrobant » les traitements métier des différents modules applicatifs (GRC, Gestion des Prestations, Gestions des Procédures, Gestion Documentaire, etc.). Exemple : le service IHM getCustomerListShortInfo() qui renvoie une liste d'usagers en 4 colonnes prêtes à l'affichage, et les traitements métier getCustomerList() et getCustomerInfo(), plus généraux. Les services IHM sont communs aux différents canaux d'accès (Web, 3G, guichet). Seuls les écrans pourront se différencier, par exemple client lourd de type WinForm pour les guichets, pages Web pour l'Internet.

Gestion des échanges avec les partenaires (couche interopérabilité)
Les échanges avec les partenaires seront au format XML, préconisation du cadre commun d'interopérabilité de l'ATICA[47]. La sécurisation des échanges se fera par utilisation de SSL V3 et de certificats X.509, toujours conformément aux préconisations de l'ATICA. Le GNU utilisera un tiers externe pour la délivrance des certificats, la vérification des certificats des autres partenaires, etc., l'interface avec cet ASP se faisant par le protocole XKMS.
Bien que l'ATICA recommande de se baser sur ebXML pour l'interopérabilité des processus collaboratifs entre administrations, pour des questions de délai, l'architecture pourra se baser sur des normes plus opérationnelles comme BPEL4WS pour l'orchestration de Web Services. Des outils de mapping permettront de s'interfacer aux télé-procédures existantes dans d'autres standards (EDI notamment).

Gestion documentaire (couche traitement)
L'application de gestion documentaire est transverse aux différents modules applicatifs du GNU. Elle exposera ses services sous forme de Web Services en utilisant la recommandation « SOAP with Attachments » du W3C pour inclure des documents binaires aux requêtes SOAP (pièces justificatives scannées, etc).
L'application de gestion documentaire devra également prendre en compte les problématiques d'habilitation de l'utilisateur ou de l'agent guichet en se basant sur les services du socle technique, la gestion de l'historique des documents, l'archivage des documents…

Gestion du Workflow (couches traitement et interopérabilité)
Le gestionnaire de Workflow sera utilisé au sein de différentes briques applicatives, notamment au sein du module Prestations et du module Procédures. Il s'agira de différentier les fonctionnalités de workflow humain (gestion des tâches et des rôles, procédures d'escalade, gestion des emplois du temps des personnes, etc.) des activités de « Process Flow » (séquencement d'envois de messages, attente de réception d'un message, etc) : Les fonctionnalités de Workflow seront réalisées par des progiciels spécifiques (exemple Akazi), le Process Flow par des progiciels d'infrastructure comme Biztalk Server, WebMethods Integrator, ou des bibliothèques Open Source implémentant BPEL4WS et du parsing XML.

47 : Agence pour les Technologies de l'Information et de la Communication dans l'Administration, www.atica.pm.gouv.fr

Le volet Développement

Ce volet a pour objectif de décrire le socle technique et méthodologique (framework) qui sera utilisé pour le développement des briques du projet.
Le volet Développement se structurera par la description des éléments suivants, en gardant comme fil directeur les différentes couches de l'architecture :

Le volet développement : des outils, des bibliothèques clé en main et des guides méthodologiques. Maximiser les deux premiers points pour pouvoir mieux limiter le dernier.

- **Les Outils** : ce sont les outils utilisés par les concepteurs pour spécifier (conception UML, intranet documentaire…), et par les développeurs lors du développement et des tests (environnement de développement, gestionnaires de sources, tests unitaires, automatisation des déploiements…). Ces outils seront spécifiés pour chacune des couches de l'architecture : Outils de développement des interfaces, IDE pour le développement des composants métier, outils de conception de bases de données, etc. C'est grâce à cette vision transverse qu'il devient envisageable de déployer dans un « build » unique des composants Java, des flux EAI, et des écrans Web.

- **Les Bibliothèques** : il s'agit d'un ensemble de fonctions ou d'objets spécifiques au projet ou à l'entreprise, et mis à disposition des concepteurs et des programmeurs. On y retrouvera des bibliothèques transverses ou spécifiques aux différentes couches de l'architecture, qu'il s'agisse de bibliothèques techniques ou métier :

  - **Librairies transverses** : structures de données complexes, gestion des traces, gestion des erreurs, des alertes d'exploitation, du multilinguisme, du paramétrage, objets métier transverses…

  - **Librairies spécifiques à la couche Présentation** : composants graphiques réutilisables (boutons, zones de saisie avec cycle de vie, menus, sélecteur de devises, fenêtre maître-détail…)
    Librairies spécifiques à la couche Traitements : montée en charge, gestion du transactionnel, émission et réceptions asynchrones, machine à état…

  - **Librairies spécifiques à la couche Persistance** : bibliothèque d'accès aux données, centralisation des requêtes, outils d'analyse d'impact…

  - **Librairies spécifiques à la couche Echanges** : outils de suivi de messages, d'ordonnancement des tâches, de parsing de messages (ex. Swift, EDI).

  - **Librairies spécifiques à la couche Sécurité** : authentification mutualisée, gestion des habilitations…

- **La Méthodologie** : il s'agit de décrire dans le détail le processus de développement de bout en bout (de la conception à l'intégration : guides de développement, patterns, méthodologie d'intégration continue…) en contraignant l'utilisation des outils et des bibliothèques dans le contexte du projet.

Le volet Production

Ce volet décrit l'architecture de production et les procédures d'exploitation associées :

- **Architecture de production** : serveurs, implantation des composants, topologie réseau, dimensionnement des machines, mécanismes de clustering, de sauvegarde et restauration, outils de supervision ou intégration à des outils de type Patrol, Tivoli…

- **Procédures** de déploiement et de configuration, procédures d'exploitation, de supervision…

La démarche

La démarche est composée de trois activités :

- La **conception de l'architecture technique**, qui se base sur les spécifications du DAG, les principaux use-case fonctionnels et les contraintes d'architecture technique à respecter (contraintes projet issues du DAG, patterns et principes d'architecture prédéfinis au niveau entreprise, etc) pour définir l'architecture technique du SI,

- La mise en place opérationnelle du **framework de développement,**

- La réalisation d'un **prototype** et l'implémentation d'un use-case permettant de valider à la fois le prototype et le framework de développement.

Comme tout édifice informatique, l'Architecture Technique se bâtit itérativement.

La démarche se voudra itérative : une première ébauche de l'AT permettra de spécifier et d'implémenter une première version du framework (outils, librairies, méthodologies), puis une première version d'un prototype permettra de valider les choix d'architecture et d'éprouver le framework de développement. Une deuxième itération permettra d'enrichir le DAT, de consolider le framework, etc, jusqu'à l'obtention d'une version du DAT et du framework jugées suffisamment stables pour servir de point de départ aux équipes de développement de la phase Implémentation.
Il est important de considérer le prototype comme l'élément central et unificateur des activités de cette phase : il permet de valider et faire évoluer simultanément l'architecture technique et le framework de développement, car on prototype à la fois une architecture ET un framework.
A chaque itération, l'activité de prototypage portera sur un use-case simple mais significatif, permettant de valider les choix d'architecture effectués pour cette itération et de dérouler, valider de bout en bout une méthodologie de développement (utilisation des outils, des librairies et des cadres d'utilisation du framework).
Enfin, rappelons que l'activité de conception, bien que déjà guidée par la structuration en couches, devra se baser autant que possible sur l'utilisation de patterns d'architecture, comme par exemple des patterns EAI[48], des patterns d'interopérabilité ou de synchronisation de référentiels comme ceux proposés par IBM[49], des patterns SSO, etc.

> **Exemple : Pattern « Gestion du transactionnel étendu entre traitements »**
> Dans un environnement distribué, nous recommandons fortement l'utilisation du gestionnaire d'échange comme gestionnaire de transactions entre traitements, sur un mode « compensation » (couplage lâche entre traitements, l'annulation de la transaction se fait grâce à un processus de compensation piloté par le gestionnaire d'échanges), plutôt que d'envisager l'utilisation de moniteurs transactionnels XA qui présuppose que l'ensemble des traitements soient « XA-compliant »…

48 : EAI Architectural patterns, Jeffrey C. Lutz, http://www.eaijournal.com.
49 : http://www-106.ibm.com/developerworks/patterns/.

Le bureau d'architecture en phase d'implémentation

L'étape précédente a permis d'identifier un palier, de finaliser un plan de mise en œuvre et de construire les fondations pour son implémentation. En théorie, tout est prêt pour aborder l'industrialisation du projet. En pratique, nous expliquons dans cette partie en quoi l'application des livrables fournis en amont (DAG, PPG, architecture technique, socle et méthodologie) nécessite un accompagnement mêlant rigueur et flexibilité.

Maîtriser l'entropie inhérente à tout projet informatique.

Bien que n'ayant pas véritablement vocation à prendre en charge la gestion du projet, l'architecte a comme engagement de maîtriser son entropie, c'est-à-dire maintenir un certain ordre dans la réalisation.

Les facteurs endogènes d'entropie proviennent essentiellement des raccourcis pris individuellement, initiatives personnelles qui, combinées entre-elles, peuvent engendrer beaucoup de complexité au niveau du projet. Les facteurs exogènes ont pour origine les nouveaux besoins fonctionnels ou les nouvelles contraintes techniques, voire des éléments technologiques d'un marché en forte maturation, pouvant apparaître au cours de la durée du projet.

Le rôle de l'architecte sera de contenir cette entropie et de gérer les événements imprévus de façon à ce que les évolutions requises restent, autant que possible, dans un cadre en cohérence avec les plans initiaux. Dans tous les cas, il s'assurera que les réalisations correspondent bien aux plans, qu'il devra éventuellement mettre à jour.

Une piste pour contenir l'entropie est de mettre en place une organisation à l'image de l'architecture, c'est à dire de découper le projet en autant de sous-projets correspondant aux briques applicatives, avec une cellule de pilotage assez réduite et des cellules projets, mixant les fonctionnels et les techniques.

Le Bureau d'Architecture du projet : une structure transverse pour encadrer et supporter les travaux de réalisation.

Pour assurer la cohérence et l'alignement de toutes ces équipes sur la stratégie globale du projet, seule une structure transverse ayant la vision globale et connaissant les tenants et les aboutissants des plans peut encadrer les travaux de spécification et de réalisation.

Ce Bureau d'Architecture du projet (BA) a une **double mission de contrôle qualité et de support**. A la croisée des interactions entre les différents acteurs du projet, son rôle peut être réparti suivant deux types de prérogatives, fonctionnelles et techniques. Le BA pourra être une émanation d'une structure centrale telle que celle décrite au chapitre Démarche d'architecture d'entreprise.

Les prérogatives fonctionnelles du bureau d'architecture

Un certain nombre de livrables ont été produits lors de la phase Etude, parmi lesquels le Dossier d'Architecture Générale (DAG), définissant les volets fonctionnels, applicatifs et techniques des plans de la cible et du palier à réaliser.

Bien qu'ayant été validé dans les grandes lignes au travers de la conception et le prototypage d'un use case représentatif, cette architecture générale n'a pas adressé de manière exhaustive et détaillée tous les cas d'utilisation et les processus à implémenter. Il faut à présent remplir les boîtes et couvrir complètement le périmètre défini.

L'action du BA, à la fois en assistance et en validation, peut se structurer autour des thèmes suivants:

- Sensibilisation et Formation,
- Maintenance et Projet,
- Interopérabilité,
- Homologation des processus transverses.

Sensibiliser à la stratégie et former aux nouveaux concepts.

Le BA doit retransmettre et expliquer aux fonctionnels les grands principes qui ont structurés les choix d'architecture, au travers de formations et autres actions de communication.

Il s'agit de les sensibiliser à la stratégie fonctionnelle motivant le projet dans son ensemble (telle que, par exemple, une orientation centrée sur le client ou le caractère stratégique d'une remontée du chiffre d'affaire à la journée) et sur la criticité de certains domaines applicatifs en particulier.

Il s'agit également de les former sur les patterns qui ont servi de base à la conception générale de l'architecture. Par exemple, le BA sera chargé de transmettre les règles qui régissent le concept de noyau et ses interactions avec les autres briques applicatives.

Faire appliquer les plans…

Le BA prend tout son sens dans son implication dans les projets. Il apporte son recul et sa vision transverse en assistant les projets pour bâtir leur dossier d'intégration dans l'architecture globale. Ainsi, il peut être amené à intervenir plus ou moins directement dans la définition et la validation des spécifications fonctionnelles d'une brique applicative, afin de garantir qu'elle ne sorte pas du périmètre prévu, qu'elle remplisse bien les fonctions pour lesquelles elle a été conçue, et que ces interactions avec les autres briques soient conformes au modèle d'échange.

Prenons l'exemple d'un SI de Distribution : la phase étude a défini une architecture dans laquelle les fonctions Paiement, Cash et Facturette ont été regroupées dans une brique Encaissement, et les fonctions Gestion du Personnel de caisse et Gestion du Catalogue Articles sont réalisées dans d'autres briques. Le rôle de l'architecte est d'assister le chef de projet Encaissement qui ne connaît pas les raisons de cette répartition fonctionnelle mais doit comprendre comment il va résoudre son besoin en termes de code Articles ou de gestion des rotations de caisses.

… ou les mettre à jour

Cette implication en profondeur du bureau architecture dans les sous-projets lui permet de conserver sa capacité d'analyse d'impact lors de l'apparition d'une nouvelle demande et d'identifier les évolutions potentielles de l'architecture. Il ne s'agit pas là de tenir à jour en temps réel une cartographie exhaustive, mais de prendre en compte avec pertinence les nouveaux besoins, afin de minimiser l'impact sur l'architecture et de garantir la « sécabilité » des briques applicatives. Lorsque des changements importants sont décidés, il s'agit de les formaliser et de les communiquer aux équipes impactées.

Pour tous ces arbitrages, le BA participe aux comités de validation des projets.

Interopérabilité : support méthodologique et gestion de la cohérence.

Sur le thème Interopérabilité, le BA fournira un support méthodologique pour la constitution des spécifications fonctionnelles des flux avec les autres briques. Il encadrera les fonctionnels sur des sujets tels que le support de transactions longues ou les impacts de la mise en place d'une gestion au fil de l'eau[50].

Il pourra imposer un formalisme commun et l'utilisation d'outils de type référentiel de flux et de contrats de service.

50 : Voir le prochain livre blanc d'OCTO Technology, à paraître, sur les méthodologies et les technologies liées à la problématique de l'interopérabilité.

Homologation des processus transverses.

Sur le thème de l'homologation des processus transverses, processus impactant plusieurs modules, le BA apportera un support en participant à l'écriture du cahier de recette et des cas de tests. Il étudiera l'opportunité d'introduire un certain niveau d'automatisation dans les tests de non-régression, par l'utilisation d'outils comme ceux proposés par un éditeur comme Mercury.

Les prérogatives techniques du bureau d'architecture

Les prérogatives techniques du BA concernent d'une part l'accompagnement autour du socle, et d'autre part l'intégration opérationnelle des briques applicatives pour assurer la cohérence bout en bout.

Le socle : des outils et des méthodologies de conception et de développement.

En dernière étape de la phase étude de la démarche, l'architecture technique a été conçue et un socle a été construit et validé au travers d'un prototype reprenant un cas d'utilisation représentatif. Ce socle va servir de cadre de travail commun (framework) pour l'implémentation de l'ensemble des briques applicatives de l'architecture.

Il a pour vocation non seulement d'implémenter les cinq couches de l'Architecture Technique, mais également de fournir les guides méthodologiques pour la conception, le développement, l'intégration et l'exploitation des briques applicatives.

Le socle contient toute l'Architecture Technique, telle que décrite dans l'étude d'architecture détaillée, selon les trois volets logiciels, développement et production. Le socle ou une partie du socle peut être issu de standards de l'entreprise mis en œuvre par des structures informatiques transverses.

L'accompagnement autour du socle : assistance, contrôle et maintenance.

En phase d'implémentation, le BA est dépositaire de ce socle. Il est chargé d'en faire la promotion, d'assister les équipes projet, et d'en vérifier la bonne utilisation, et d'en assurer la maintenance pour adresser les nouveaux besoins.

Une de ses principales actions est de former, puis d'assister, les équipes de développement à l'utilisation de l'environnement de travail du socle et des méthodes de développements à appliquer.

Un bon socle ne se juge pas au kilo
Il ne s'agit pas de noyer les projets sous des tonnes de documentations éditeurs, ni de leur imposer une somme astronomique de règles contraignantes qu'ils n'appliqueront jamais. L'objectif est de leur faciliter la tâche d'acquisition de ce socle par la fourniture de cadres d'utilisation précis et succincts. Le succès de cette démarche passe par l'adhésion des équipes. Elles doivent percevoir les gains en efficacité, qualité et flexibilité, issus de la bonne utilisation du socle. C'est pourquoi là aussi, la formation est primordiale.

Les nouvelles méthodologies de développement mettent l'emphase sur les cycles courts et la responsabilisation.

Le principal défi de cette phase est d'éduquer sur les principes de développement telles que ceux mis en avant dans les méthodologies agiles (ex. : eXtreme Programming[51]). Ces nouvelles méthodologies sont centrées sur les écueils constatés dans les projets informatiques, c'est à dire la qualité et l'aptitude à supporter le changement. Elles préconisent une collaboration étroite et une forte délégation de responsabilité aux développeurs, allant de la conception à la validation de leur code, par une utilisation systématique des tests unitaires[52].

51 : http://www.extremeprogramming.org/ ou http://www.xprogramming.com/.
52 : cf. l'ouvrage sur les tests unitaires de Vincent Massol à paraître aux éditions Manning (http://www.manning.com/)et le livre blanc d'OCTO Technology sur le développement d'applications, à paraître également.

L'autonomie de la brique, sous contrôle qualité.

Une brique est une brique. La démarche d'architecture consistant à découper en briques, elle permet de donner le maximum d'autonomie (donc de responsabilités !) aux équipes chargées de leur implémentation ; mais à une condition : que cette implémentation respecte les fondations communes. En conséquence, le BA peut notamment opérer des revues de codes ou de tests unitaires pour vérifier la qualité des développements. Le cas échéant, lorsque le framework a été contourné sans son accord, il pourra exiger le re-factoring du module concerné.

Exploitation bout en bout par un monitoring technique et applicatif.

En bout de chaîne, le BA sensibilisera les équipes de production à l'exploitation de bout en bout, en mettant notamment en place des outils et des procédures de gestion d'alertes applicatives. L'objectif est de responsabiliser ces populations, en plus de l'exploitation des ressources techniques (réseaux, machine, etc.), à un niveau d'exploitation applicatif, tels que par exemple le dépassement de seuil haut ou bas de files d'attente, ou l'allongement des temps de réponse. Une collaboration entre le BA et les équipes de production sera à privilégier.

Maintenance corrective et évolutive de l'infrastructure technique.

Quelque soit la taille ou la durée du projet, un travail de maintenance des infrastructures est à prévoir. D'une part, les bugs techniques ou les lacunes fonctionnelles détectés par les utilisateurs (les équipes de développement, d'intégration et d'exploitation) devront être corrigés. D'autre part, les évolutions fonctionnelles ou les innovations technologiques donneront lieu à de la maintenance évolutive sur le socle.

Au sein de l'équipe du BA, des experts techniques de chaque couche s'assureront, lors de la phase d'initialisation et tout au long de la vie du projet, du « bétonnage » du socle en termes notamment de sécurité, de performance, de tolérance aux pannes et d'évolutivité. Ces experts seront régulièrement sollicités pour la résolution des problèmes, le passage des patchs éditeurs, voire la montée de version des produits utilisés dans le socle.

Une des dernières prérogatives techniques (« last but not least », comme le veut la formule consacrée) est la responsabilité de la plate-forme d'intégration.

Responsabilité de la plate-forme d'intégration.

Comme nous avons pu déjà l'évoquer à plusieurs reprises dans ce document, le principe de la démarche d'architecture Projet est de d'éclater une problématique en plusieurs briques et de réintégrer leurs implémentations dans une solution globale et homogène répondant au besoin exprimé.

En collaboration avec les équipes applicatives du BA, les équipes techniques doivent donner les moyens aux briques de tester leurs fonctionnalités indépendamment de la disponibilité des autres briques, puis de les intégrer aux autres briques.

Il faut par conséquent d'une part leur fournir des « injecteurs » et des « bouchons » simulant les interactions externes, et d'autre part disposer d'un environnement d'intégration permettant la recette progressive des briques par la mise en œuvre des processus transverses : passage des tests de bout en bout et construction des packages constituant la solution globale.

Conclusion

S'il n'intervient pas directement dans l'implémentation des briques applicatives, le BA est pourtant omniprésent, car responsable du respect de l'architecture sur laquelle il s'est engagé en phase Etude.

Plus il sera pédagogue et convaincant en assistance, meilleure sera la qualité des réalisations, et par conséquent moins il aura à exercer de rôle coercitif.

La formation sur site est un élément clé de la maîtrise de la qualité.

Comme toute activité industrielle, l'informatique s'est attachée dans ses premières heures à produire vite, sans souci réel de qualité, de développement durable, et de beauté finalement. Lorsque Michel Pébereau parle de Cathédrales à propos de nos Systèmes d'Information, son lyrisme travestit une réalité constituée d'usines encombrées plutôt que d'édifices gothiques. Car à l'image des usines industrielles du milieu du siècle dernier, où se trouve le beau ?

Cette notion de beau a fait son chemin dans l'entreprise pour améliorer globalement les conditions de travail. L'architecture de Systèmes d'Information est quant à elle une discipline jeune, qui a motivé la création d'OCTO. Pour les mêmes raisons, peut-être nous conduira-t-elle jusqu'à des activités de design, des « architectures signées Stark », ou comme me le soufflait un client, « OCTO des architectures à vivre ».

VI. Bibliographie

[1] **Nouvelle économie** : rapport Daniel Cohen et Michèle Debonneuil - Paris : La Documentation française, 2000. 251 p. (Collection Les Rapports du Conseil d'analyse économique ; 28).

[2] Frédéric Fréry : « **Benetton ou l'entreprise virtuelle** », Vuibert, 1999.

[3] « **Blown into Bits : How the New Economics of Information Transforms Strategy** » (ou comment la nouvelle économie de l'information transforme la stratégie), de Philip Evans et Thomas Wurster, publié par Harvard Business School Press courant 1999.

[4] Les Echos – **Un système d'Information pour être compétitif**, de Peter Weill et Marianne Broadbent.

[5] Les Echos – **Les organisations du XXIe siècle**, de Marc Lemarignier et Jessica Scale.

[6] **Urbanisation du business et des SI**, de Gérard Jean (éd. Hermès).

[7] **Le projet d'urbanisation du système d'information**, de Christophe Longépé ISBN 2-100-05694-8 Dunod.

[8] **Evaluating Software Architectures** – Methods and Case Studies Paul Clements, Rick Kazman, Mark Klein ISBN 0-201-70482-X202 Addison-Weisley.

[9] **Le processus unifié de développement logiciel** - Ivar Jacobson, Grady Booch, James Rumbaugh ISBN 2-212-09142-7 Eyrolles.

[10] **UML en action Pascal Roques**, Franck Vallée ISBN 2-212-09127-3 Eyrolles.

[11] **La société de la connaissance** - Jean-Pierre Corniou, Hermès.

[12] **Design Patterns**, Elements of Reusable OO software, The Gang of 4.

[13] **Pattern Oriented Software Architecture**, Editions Wiley & Sons.

[14] **The 4+1 view model of software architecture**, IEEE 1995.

[15] France Télécom, **Mémento Technique N°14** (www.rd.francetelecom.fr).

[16] **EAI Architectural patterns**, Jeffrey C. Lutz.

VII. A propos d'OCTO Technology

www.octo.com

OCTO Technology est le cabinet de conseil en Architecture de Système d'Information du groupe Aubay. Depuis sa création en 1998, OCTO Technology publie les fruits de ses expériences sous forme de livres blancs. Ils sont diffusés et diffusables gratuitement.
Après avoir publié autour des trois offres « technologiques » : les serveurs d'applications et IRM, l'EAI, et la Sécurité, OCTO Technology vous propose ce dernier Livre Blanc, qui synthétise son savoir faire en terme de construction d'architectures complexes. Dans ce domaine, ses clients sont des acteurs grands comptes, soucieux d'exploiter au mieux les nouvelles technologies, et considérant l'architecture comme un enjeu stratégique pour leur SI d'aujourd'hui et de demain. Citons parmi eux : Air Liquide, BNP Paribas, Natexis, Société Générale, France Télécom, Total Fina Elf, JC Decaux, Vivendi, Orange, Wanadoo ...

Auteurs :

Les auteurs de ce document sont des consultants d'OCTO Technology, par ordre alphabétique : Laurent Avignon, Gilles Laborderie, Rémy Mathieu-Daudé, et Pierre Pezziardi.

Remerciements spéciaux à Eric Groise sans qui ce libre blanc n'aurait jamais vu le jour, et à Lyonel Thouvenot pour la qualité visuelle du produit fini.

Pour tout complément d'informations : architecture@octo.com

Bibliographie OCTO Technology

Le Livre Blanc des Serveurs d'Application © OCTO Technology 1999
Serveurs d'Applications (éd. Eyrolles)
Le Livre Blanc de l'EAI© OCTO Technology 1999
Intégration d'Applications (éd. Eyrolles)
Le Livre Blanc de l'IRM © OCTO Technology 2000
Le projet eCRM (éd. Eyrolles) - lauréat du prix AFISI 2002
Le Livre Blanc de la Sécurité © OCTO Technology 2001
Architectures de Systèmes d'Information, Livre Blanc © OCTO Technology 2002

Dépôt légal : Décembre 2002
Imprimé pour OCTO Technology chez
PORTAPRINT - 2 rue du Blanc-Seau
59334 - TOURCOING CEDEX
N° de dosier : 2002110026

OCTO Technology - 62 rue La Boëtie - 75008 Paris
Tél : (33) 1 58 56 10 00 - Fax : (33) 1 58 56 10 01 - http://www.octo.com