Approche de l'aspect logique

Sujet	**Dimension « Produit »**
Objet du Guide	Définir l'aspect logique du Système Entreprise et décrire son contenu. La modélisation logique, à l'appui de l'architecture et de la conception logiques, établit l'architecture du système technique, et spécifie formellement ses composants.
Mots clefs	aspect logique, architecture, système, informatique, modélisation, Produit, UML

Référence	PxPRD-50	*État*	En cours de rédaction
Version	1.0.0	*Date*	29 février 2020
Auteurs, contributeurs	Dominique VAUQUIER, contribution d'Emmanuel REYNAUD		
Relecteurs	Pierre BONNET, Thierry BIARD		

Sommaire

Rappels méthodologiques

Le corpus méthodologique de Praxeme est constitué de :

- Guides, qui fournissent les fondamentaux d'un sujet ou d'un domaine ;
- Procédés, définis comme des « façons de faire, modes opératoires pour exécuter une tâche »[1] ;
- Processus, qui décrivent des enchaînements d'activités et les dispositions d'organisation qui les accompagnent.

En dehors de la méthode, le fonds public Praxeme contient des modèles et des matériaux pédagogiques.

Pour suivre l'actualité de la méthode publique

- Mailing list
- Groupe LinkedIn

Voir http://www.praxeme.org

- Le wiki : http://wiki.praxeme.org

Pour contribuer aux travaux du Praxeme Institute

mailto:info@praxeme.org

Protection du document

L'initiative pour une méthode publique repose sur le bénévolat et la mutualisation des investissements entre ses contributeurs. Elle vise à élaborer et à diffuser une méthode ouverte et libre de droits. Sa dynamique n'est possible que si cet esprit est maintenu à travers les utilisations des documents qu'elle met à la disposition du public. C'est pourquoi les documents sont protégés par une licence « *creative commons* »[2] qui autorise l'usage et la réutilisation de tout ou partie d'un document du fonds Praxeme sous seule condition que l'origine en soit citée. Les éventuels documents dérivés, qui reprennent du contenu de Praxeme, doivent s'appliquer à eux-mêmes les mêmes conditions, faire référence à la « *creative commons* » et porter les symboles idoines :

Actualisation de ce document

Pour obtenir la dernière version de ce document, se rendre sur le site web du *Praxeme Institute*, à la page : http://www.praxeme.org/telechargements/catalogue/.

L'historique du document

Indice	Date	Rédacteur	Contenu
0.0.0	02/02/2020	DVAU	Première rédaction
0.1.0	20/02/2020		Après retour d'Emmanuel REYNAUD
0.1.1	21/02/2020		Retours Thierry BIARD
1.0.0	29/02/2020		Après retours ER, TBIA, Pierre BONNET
1.0.0	29/02/2020		Version actuelle du document

[1] Cf. rubrique Thesaurus sur le site du *Praxeme Institute* : http://wiki.praxeme.org/index.php?n=Thesaurus.Procedure.

[2] Voir la philosophie et le détail des licences sur : http://creativecommons.org/.

Toute méthode consiste, au fond, à bien isoler et connaître ses éléments – le reste n'est rien, il se fait tout seul.

Paul Valéry, Cahiers, tome I, p. 778, édition La Pléiade

1. Introduction

1.1 Définition de l'aspect logique

Nous percevons l'entreprise comme un système complexe. Un tel système ne peut être appréhendé qu'à travers plusieurs aspects, à la fois distincts et soigneusement articulés. La Topologie du Système Entreprise est la grille de lecture que Praxeme propose pour appréhender la réalité de l'entreprise. Elle recense et ordonne les aspects de l'entreprise[3]. Parmi ces aspects, l'aspect logique s'intercale entre les aspects dits « métier » et les aspects portant sur les systèmes techniques au service de l'entreprise. Dans cette position intermédiaire, il joue un rôle essentiel pour concevoir les solutions techniques et garantir leur alignement sur les besoins de l'entreprise.

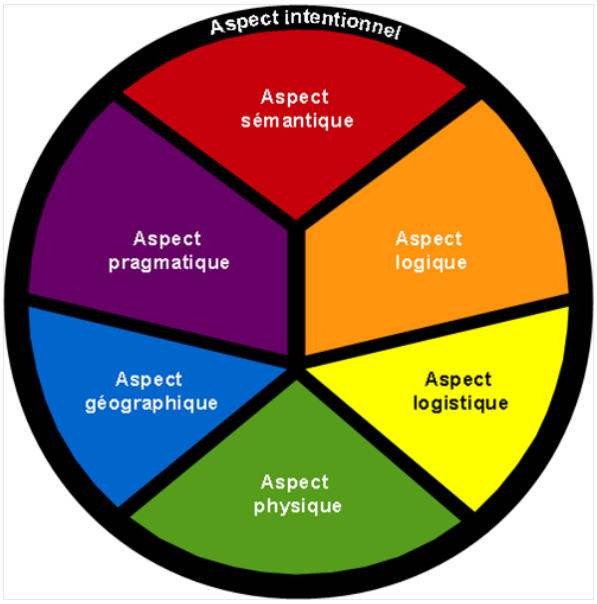
L'aspect logique d'un système fournit une abstraction de ses moyens logistiques et techniques.

Cette notion n'est pas nouvelle : depuis l'aube du génie logiciel, les méthodes ont toujours identifié un « niveau » logique. Par exemple, on sait ce qu'apporte un modèle logique des données. Il ne décrit pas les informations du point de vue des acteurs « métier », mais il peut se référer à un modèle des connaissances (modèle conceptuel). Il ne se confond pas avec le schéma physique de la base de données, mais il l'anticipe ou le spécifie à partir du choix de la technique de persistance (par exemple, base de données relationnelle). L'aspect logique généralise, à l'ensemble du système, ce regard, intermédiaire entre métier et technique.

À travers l'aspect logique, c'est bien le système technique ou informatique qui est décrit, mais abstraction faite des choix technologiques précis. De ce fait, la description logique, protégée des changements techniques, jouit d'une généralité plus grande et d'une espérance de vie supérieure, par rapport à la description technique.

1.2 Positionnement de l'aspect logique

Figure PxPRD-50_1. La Topologie du Système Entreprise



Le système peut être une entreprise, un dispositif technique ou sociotechnique, un « système produit » (système de transport, système d'armement…), toute combinaison de moyens humains et matériels en vue d'une finalité explicite.

Les éléments conceptuels et organisationnels qui interviennent dans ce système sont abordés à travers les aspects sémantique et pragmatique. Les moyens matériels et techniques s'assemblent dans les aspects logistique et physique. La Topologie du Système Entreprise intercale un aspect intermédiaire, l'aspect logique, entre le métier et ses moyens logistiques.

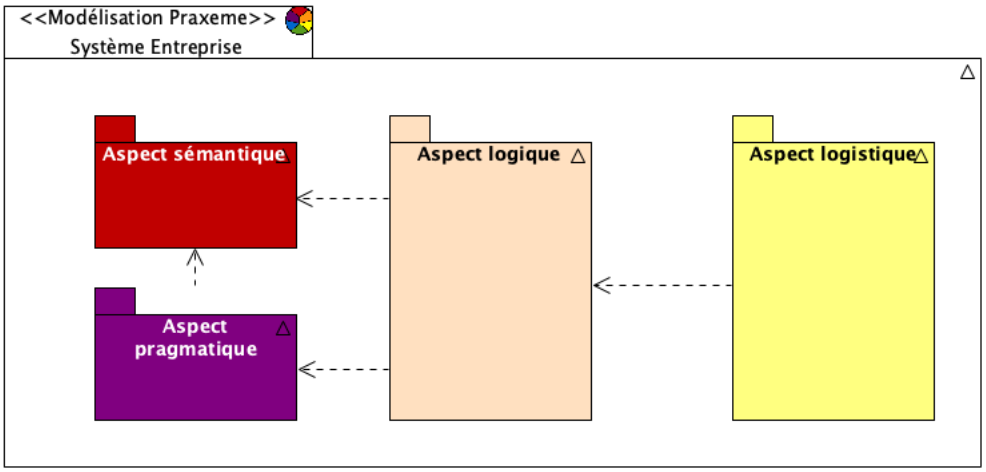
L'aspect logique s'insère, donc, comme un intermédiaire entre les premiers – aspects « métier », en amont – et les derniers – « aspects techniques », en aval.

La description logique établit les plans du ou des systèmes techniques à réaliser.

[3] Voir le guide « La Topologie du Système Entreprise », réf. PxPRD-01.

En conséquence, le travail à mener sur cet aspect bénéficie de deux points de départ : l'aspect sémantique et l'aspect pragmatique, ou, si l'on veut : les « objets métier » et les « activités métier ». Cette position marque un changement par rapport à la tradition méthodologique. Classiquement, les niveaux se succèdent dans la démarche : conceptuel, puis organisationnel, puis logique. Ceci justifie, d'ailleurs, le terme de « niveau ». Praxeme modifie cette séquence, et détermine l'architecture logique à partir de deux aspects amont. Ce changement produit une transformation considérable : il se lit dans la physionomie des systèmes informatiques.

Figure PxPRD-50_2. Le voisinage de l'aspect logique



Ce diagramme est un extrait de la Topologie du Système Entreprise. Il ne montre que les aspects contigus à l'aspect logique. Les aspects sont représentés par des paquetages, notion UML qui offre un mécanisme de rangement. Ces paquetages accueillent des éléments de modélisation, dont la nature précise est propre à chaque aspect.

Sur ce diagramme, les flèches en pointillés explicitent les dépendances entre les aspects. Elles résument des dizaines de règles de dérivation qui guident le concepteur. Cette notion de dérivation sera détaillée plus loin.

a. En amont : l'aspect sémantique

L'aspect sémantique est ainsi en contact avec l'aspect logique. C'est ainsi que la méthode traduit l'idée qu'une partie au moins des solutions techniques devrait être conçue en référence aux fondamentaux du métier. Cette idée, énoncée ainsi, devrait paraître évidente. Pourtant, les pratiques courantes sont loin de la respecter. Historiquement, l'approche fonctionnelle a façonné un esprit fort différent, entraînant des conséquences désastreuses dans les systèmes existants.

Dans le style orienté services, la question clef « Comment trouver les services ? » trouve une partie essentielle de sa réponse dans l'exploitation du modèle sémantique.

b. En amont : l'aspect pragmatique

Les habitudes de travail, les règles d'organisation, les rôles et les activités associées forment également la réalité de l'entreprise. Il s'agit d'un autre point de vue sur le métier, certes contingent, mais également nécessaire pour concevoir le meilleur équipement possible. Le modèle pragmatique, c'est-à-dire la description de ces éléments liés à l'organisation, doit aussi être pris en compte dans l'architecture des systèmes techniques. Ces éléments étant d'une nature différente par rapport aux éléments sémantiques, d'autres règles de dérivation s'appliquent pour obtenir d'autres catégories de constituants logiques.

c. En aval : l'aspect logistique

L'aspect logistique contient les types de moyens matériels (véhicules, machines, ordinateurs…), ainsi que les composants logiciels qui les contrôlent. Ils relèvent de décisions de conception propres, lesquelles s'appuient sur l'expression logique et la traduisent dans les termes techniques retenus.

Alors qu'en principe, l'aspect logique se définit comme indépendant des choix techniques, la relation entre l'aspect logique et l'aspect logistique est, toutefois, un peu plus compliquée. Elle apparaîtra dans la notion de style et appellera un moment particulier dans la chaîne de transformation : la négociation logique-technique.

1.3 Enjeux de l'approche logique

Du fait du déclin des pratiques et compétences sur ce niveau logique, à travers le marché, il est sans doute nécessaire de revenir sur les enjeux et les retombées que l'on peut attendre en restaurant l'approche logique des systèmes.

a. Maîtriser l'évolution des systèmes techniques

La transformation des systèmes techniques, particulièrement des systèmes informatiques, peut être rendue nécessaire par leur état, par l'évolution des besoins, comme par la disponibilité de nouvelles solutions techniques.

La plupart des systèmes informatiques actuels présentent un taux de redondance effarant, un couplage mal maîtrisé et des défauts de documentation. Cette situation est lourde de risques, à un moment où l'outil informatique devient de plus en plus déterminant pour la qualité des services rendus par l'entreprise et sa réactivité dans un environnement changeant.

Or, la transformation des systèmes techniques prend du temps. Même avec une continuité de vision et un management volontariste, elle se déroule sur plusieurs années. Pendant ce temps, l'architecture technique peut changer plusieurs fois. Si nous ne disposons que d'une description technique du système, cette description étant instable, nous aurons du mal à guider la transformation.

Un des avantages de la description logique est son découplage par rapport aux choix techniques. Grâce à ce découplage, elle reste valable pendant plus longtemps et peut accompagner des transformations significatives du système.

De plus, le temps long de la transformation requiert l'implication au plus haut niveau de l'entreprise. Il est donc nécessaire de présenter un plan de la transformation qui soit communicable à des décideurs n'ayant pas la culture technique. La description logique joue ce rôle. Plus abstraite que la description technique, elle est aussi plus facile à communiquer. Quelques graphes d'architecture permettront de faire comprendre le contenu et la raison de la transformation.

b. Améliorer la qualité des systèmes techniques et réduire leurs coûts

La qualité des systèmes s'analyse en facteurs et critères. Parmi les facteurs envisageables, les paragraphes suivants se penchent sur l'agilité et l'interopérabilité. Une composante importante de la qualité des systèmes est la qualité structurelle. Elle se mesure en termes de couplage, autonomie, redondance.

Un système bien conçu est faiblement couplé et non-redondant.

Ces notions et les métriques qui les accompagnent nous viennent de l'approche d'analyse-conception structurée. Elles restent tout à fait pertinentes dans le cas des architectures actuelles. Elles s'appliquent aux systèmes existants, et pourraient être évaluées en analysant l'architecture applicative, c'est-à-dire les composants logiciels. Mais il est rare de consentir cet effort. Qui plus est, l'architecture du logiciel prend en compte des phénomènes qui lui sont propres. Par exemple, une même fonction pourrait, fort légitimement, déboucher sur deux réalisations informatiques parce qu'elle se déploie sur deux cibles techniques. On a donc à faire à une sorte de redondance. De même, au niveau physique, l'architecte peut très bien décider de doubler certains dispositifs (par exemple, une base de données) pour des raisons de performance ou de sécurité. Une telle décision est tout à fait fondée ; elle n'est pas contraire à l'exigence de qualité.

Nous devons donc préciser ce que nous entendons par qualité structurelle. Pour l'appréhender, nous devons gommer les complications qui naissent aux niveaux technique et physique. C'est donc sur l'aspect logique que cette analyse est menée. L'architecture logique révèle les dépendances entre les constituants et donne une idée précise de leur contenu. À partir de là, nous pouvons évaluer la qualité structurelle du système.

Le paragraphe 2.4c, p. 23, précise les définitions liées à la qualité structurelle. La responsabilité de l'architecte logique est, avant tout, d'optimiser le système à construire, en commençant par assurer sa qualité structurelle.

L'élimination de la redondance et la réduction du couplage ont un impact économique direct sur le développement et le fonctionnement d'un système.

En moyenne et en première approche, on peut considérer que les systèmes informatiques sont redondants à 50%. Éliminer cette redondance permettrait de diminuer d'autant les budgets informatiques et de rendre les évolutions plus rapides et moins coûteuses. Une telle transformation résolue repose sur une vision précise que seule l'architecture logique peut apporter.

c. Contribuer à l'agilité de l'entreprise

L'agilité est la capacité à s'adapter rapidement et à moindre coût. L'entreprise en a besoin pour s'adapter aux changements, positifs ou négatifs, de son environnement. D'une part, elle doit éliminer tout ce qui l'alourdit, la freine, l'empêche ; d'autre part, elle veut saisir les opportunités de transformation qui se présentent : évolution du marché, changement réglementaire, possibilités technologiques, reconfigurations organisationnelles, etc.

Trop souvent, ces changements sont ralentis, voire interdits, à cause du poids et de la complication des systèmes techniques. Un cas typique est la programmation « en dur » et la distribution des règles métier, saupoudrées et perdues au sein du patrimoine applicatif.

Assurer l'avenir de l'entreprise demande donc :

1. d'éliminer les surpoids ;
2. de repérer les points de variation, pour y répondre à l'aide de dispositifs d'agilité.

Le premier point renvoie à la discussion sur la qualité structurelle. Le deuxième évoque des solutions techniques, mais qu'il faut positionner soigneusement dans l'ensemble du système. Dans les deux cas, le travail doit être mené en termes logiques.

d. Assurer l'interopérabilité des systèmes

La problématique, aujourd'hui, ne se limite pas à un système (une entreprise avec un système informatique…). Nous avons affaire à des fédérations de systèmes : un ensemble d'entreprises liées par des relations juridiques, contractuelles, organisationnelles, opérationnelles… À l'intérieur de chacune, plusieurs sous-systèmes coexistent (filiales, silos organisationnels et/ou applicatifs…). La performance dépend de la bonne circulation des informations et de la coordination des actions, à tous les niveaux et dans toutes les composantes de ce multi-système.

Dès lors, l'interopérabilité s'impose comme une préoccupation majeure. À l'origine, elle compte parmi les plus puissantes motivations de l'approche SOA. La technologie apporte son lot de solutions, protocoles, standards, pour contribuer à l'interopérabilité. Encore faut-il lui trouver un contenu : langage pivot, interfaces, services exposés… Pour bien concevoir ces éléments, il est nécessaire de concilier le point de vue global, multi-système, et l'attention aux détails. Par exemple, le langage pivot doit être conçu pour qu'il convienne, sans effort, à un maximum d'acteurs impliqués dans la fédération de systèmes. Les bonnes pratiques pour y arriver sont définies au niveau logique.

Pour rester à la hauteur de cet enjeu, la méthode ne doit pas se borner à la conception d'un système, mais, d'emblée, elle doit se montrer capable d'embrasser un ensemble de systèmes et de prescrire une approche multi-système.

e. Gérer les compétences

Incidemment, la bonne gestion des compétences devrait préoccuper les directions informatiques et les équipes d'ingénierie. Les compétences techniques – c'est-à-dire la programmation, l'expertise technique, l'exploitation, etc. – se périment au même rythme que les choix techniques se renouvellent. Par exemple, la maîtrise d'un langage de programmation est une vraie compétence, mais sa valeur dépend de la popularité de ce langage sur le marché. Quand la présence de ce langage régresse, se pose la question de la conversion de la compétence.

De même, l'expertise technique, très difficile à acquérir, portant sur une solution, aura une valeur tant que cette solution n'est pas détrônée par une nouvelle.

Il faudra toujours entretenir un vivier pour porter ces compétences techniques. Cependant, il est possible de réduire le coût d'entretien de capital intellectuel. Il suffit de considérer que les compétences sur l'aspect logique sont plus pérennes : elles ne changent que lors des changements de paradigmes, c'est-à-dire rarement. Ainsi, si les algorithmes sont élaborés directement par les développeurs, donc dans un langage de programmation, au moment où ce langage sera remplacé par un autre, les algorithmes seront perdus et il faudra former les développeurs à un nouveau langage. Au contraire, les concepteurs logiques peuvent élaborer les algorithmes dans des termes indépendants des langages de programmation (pseudo-langage). Non seulement leurs livrables ne seront pas affectés par les changements techniques, mais surtout leur compétence y survivra.

Les compétences sur l'aspect logique sont plus stables que les compétences techniques. Il y a donc un intérêt économique à leur ménager une place dans les activités de construction des systèmes techniques.

1.4 Objet et contenu de ce guide

Ce document décrit l'aspect logique, son contenu, les règles de l'art pour le représenter et pour concevoir des systèmes techniques optimaux. En tant que guide de la dimension « Produit », il ne s'occupe que de la substance de l'aspect, pas de la façon de l'aborder. Il introduit les procédés, décrits par ailleurs (voir en conclusion).

L'aspect est d'abord approché en tant que tel : le chapitre « Contenu de l'aspect logique » analyse sa substance et fixe les catégories sous lesquelles cet aspect nous apparaît. La notion de style est cardinale, dans cet aspect. Le choix d'un style détermine la manière d'envisager le système.

Le chapitre « Description de l'aspect logique » introduit les techniques de représentation adaptées à cet aspect.

Enfin, le dernier chapitre présente les utilisations des modèles logiques dans les démarches de transformation.

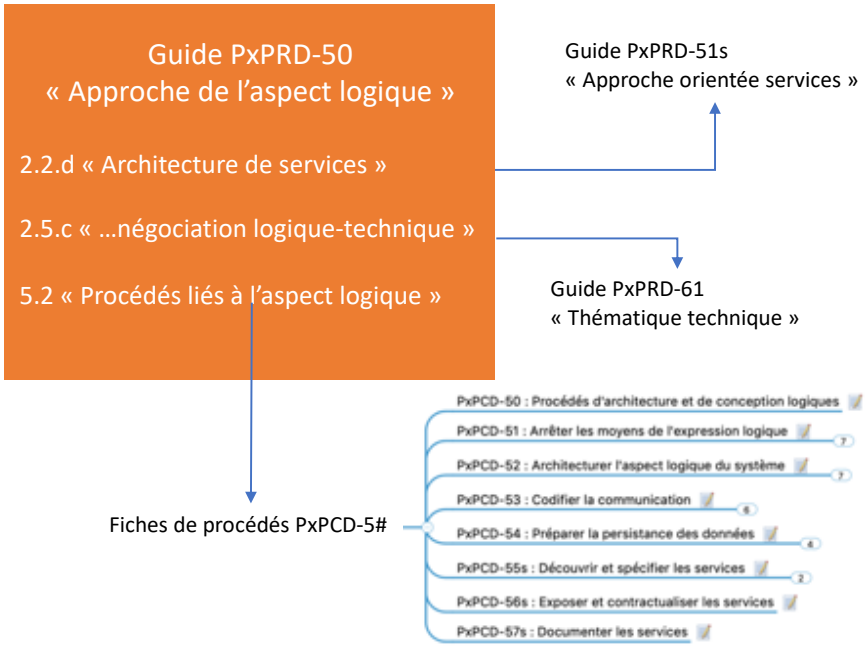
Ce document cherche à fonder les pratiques d'architecture logique et de conception logique, sachant les enjeux auxquels elles répondent. Pour mieux préparer l'avenir, il se veut indépendant d'un style particulier. Cette position conduit à dégager les notions et préceptes essentiels à ces disciplines. Le prix à payer est l'élimination des précisions qui adhèrent à tel ou tel style. Afin d'appuyer les pratiques immédiates, le présent guide s'accompagne de guides spécialisés sur un style particulier.

Le guide « Approche de l'aspect logique » pose les bases universelles des disciplines d'architecture et de conception logiques, valables quel que soit le style d'architecture retenu pour le système technique.

Cet effort d'abstraction peut rendre l'accès plus difficile.

La figure suivante récapitule les documents de la méthode qui ont une relation avec le guide de l'aspect logique. La lettre 's' à la fin de l'indice du document indique que le contenu s'applique au style orienté services.

Figure PxPRD-50_3. La structure de la documentation pour la méthode de l'aspect logique



2. Contenu de l'aspect logique

Chaque aspect identifié dans la Topologie du Système Entreprise est un espace d'expression qui offre un langage précis, utilisé pour décrire une des composantes de l'entreprise. Le langage fournit :

1. un vocabulaire, pour nommer les éléments de cet aspect ;
2. une syntaxe, ensemble de règles permettant de relier ces éléments.

Ce chapitre présente le vocabulaire, les types d'éléments que l'on trouve dans l'aspect logique.

2.1 Constituants logiques

La notion la plus générale est celle de « constituant logique ». C'est ainsi que nous nommons les éléments avec lesquels nous construisons les systèmes, vus sous leur aspect logique. Ces constituants logiques sont représentés par des éléments de modélisation, dans des modèles logiques[4]. Un modèle logique décrit tout ou partie de l'aspect logique d'un système.

Dans le cas d'un système informatique, les constituants logiques fournissent la spécification des composants logiciels à développer.

Dans le cas d'un « système produit », ils dessinent, au premier niveau de décomposition du système, des sous-systèmes, lesquels assemblent du matériel et du logiciel. Par exemple, pour un système de transport ou un système d'armement, l'architecture logique découpe des sous-systèmes qui correspondent à des véhicules, des armes, des stations de contrôle… reliés par des dispositifs de communication et dotés d'intelligence logicielle.

Une conception logique qui se voudrait innovante se garderait bien de reproduire trop vite la décomposition des systèmes physiques en place. Elle partira de la description la plus abstraite, à rechercher dans les aspects sémantique et pragmatique, pour se laisser la liberté d'imaginer une autre architecture.

2.2 Styles d'architecture logique

Nous nous sommes habitués à l'usage métaphorique du terme « architecture », au point d'occulter son contenu. La véritable architecture, l'art de concevoir et construire des édifices, obéit à des règles – les « règles de l'art », la réglementation – et commence par le choix, tacite ou explicite, d'un style. La plupart du temps, en retenant

[4] La question de la représentation fait l'objet du chapitre 3.

l'architecte, le commanditaire choisit le style. Pour construire un édifice satisfaisant un cahier des charges et réalisant une même fonction donnée, il y a énormément de possibilités, tant matérielles que formelles et esthétiques. Entre le fonctionnalisme des années 70 et le gaspillage baroque d'un Frank Gehry, le spectre des possibilités est large. Une des premières décisions à prendre consiste à resserrer cet éventail de possibles. Le choix du style y contribue.

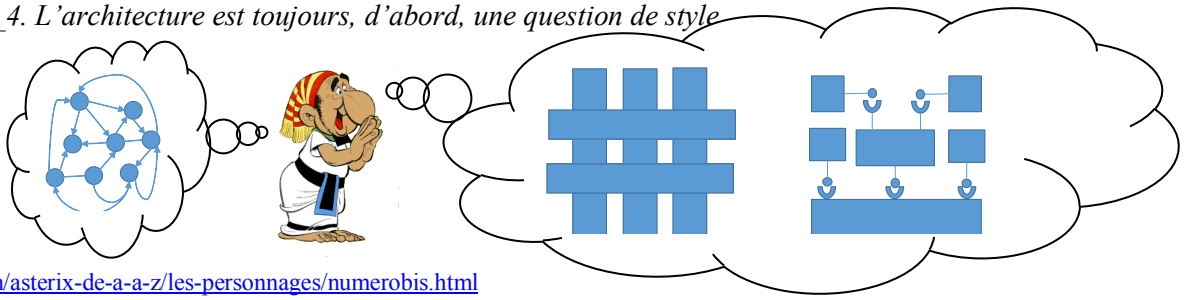
a. Notion de style

Le besoin est le même quand on aborde l'architecture de l'entreprise. Plus précisément, l'architecture logique focalise la réflexion sur les systèmes techniques aux services de l'entreprise. Son statut intermédiaire débouche sur un certain arbitraire dans son expression comme dans ses règles : elle ne parle plus de la réalité « métier », pas encore de la technologie. Il faut donc lui trouver un vocabulaire. De fait, les méthodes ont toujours recouru à la métaphore pour proposer un vocabulaire applicable à l'aspect logique, par exemple :

- machine logique et serveur logique de la méthode TACT ;
- urbanisation des systèmes d'information ;
- fabrique, composant… et autres termes tirés de la métaphore industrielle dans l'approche par les composants ;
- service…

Le choix de la métaphore n'est pas neutre : il infère une certaine perception du système et de sa substance ; il renvoie, toujours, à un style précis.

Figure PxPRD-50_4. L'architecture est toujours, d'abord, une question de style



© Goscinny & Uderzo,
http://www.asterix.com/asterix-de-a-a-z/les-personnages/numerobis.html

Un style d'architecture logique se caractérise par :

1. les types de constituants logiques, nommés en filant la métaphore,
2. les principes de structuration qui permettent de les assembler (ou, pour le dire autrement, la façon de décomposer le système),
3. les modes de communication entre ces constituants.

La notion de style d'architecture logique renvoie à celle de paradigme.

> *Un paradigme est un ensemble de croyances et de principes partagés par une communauté professionnelle ou scientifique*[5].

[5] Dans cette acceptation, nous devons ce terme à l'épistémologie, plus précisément à Thomas Kuhn, dans son ouvrage *La Structure des révolutions scientifiques* (1962). Il y montre que le progrès de la science, contrairement à une idée reçue, ne procède pas par accumulation, mais par rupture. Chaque rupture correspond à un changement de paradigme. Un des exemples est l'avènement de la relativité générale, quand la mécanique newtonienne s'est montrée décidément incapable de rendre compte d'observations comme la trajectoire de Mercure ou la déviation des rayons lumineux. Toute proportion gardée, la façon de percevoir et concevoir un système (entreprise, produit, informatique…) s'analyse bien en termes de paradigmes pour mettre en évidence les schémas de pensée qui conditionnent notre aperception et nos décisions.

En informatique, un exemple de paradigme est l'approche classique, essentiellement fonctionnaliste et reposant sur le postulat de la séparation donnée-traitement. Abandonner ce postulat, c'est un peu comme retirer l'axiome des droites parallèles et passer de la géométrie euclidienne à la géométrie de Riemann.

Tout paradigme informatique repose sur un état de la technologie. Il y a adéquation entre les notions et principes structurants, exprimés en système dans le paradigme, d'un côté, et les termes techniques des solutions, de l'autre.

La différence entre paradigme et style se trouve dans l'application. Le paradigme conditionne notre mode de raisonnement et – antérieurement – de perception. Le style dont nous parlons ici est une application consciente d'un paradigme pour la conception d'un système. Cette notion vaut donc dans l'aspect logique et appelle une décision d'architecture logique.

Tous les paradigmes ne représentent pas de bons candidats pour forger un style. Par exemple, l'approche orientée objets, malgré ses mérites, ne peut pas s'appliquer à l'échelle d'un système informatique, en tout cas pas dans l'état actuel de la technologie.

b. Choix d'un style pour l'aspect logique

Choisir un style d'architecture logique n'est pas une affaire d'esthétique, encore que, comme en mathématiques, la qualité structurelle du système peut être perçue en termes d'élégance. Le choix du style dépend des facteurs suivants :

- la génération des technologies visées,
- les compétences de conception disponibles,
- les orientations stratégiques.

En effet, la description logique a pour finalité d'être convertie en termes techniques, pour réaliser un système technique qui fonctionne. Même si elle se veut découplée ou indépendante des choix techniques, elle doit pouvoir se traduire, techniquement, sans trop de distorsion. C'est ainsi que les méthodes des années 70 et 80 reposaient sur le postulat de la séparation données-traitements, puisqu'il caractérisait la technologie de cette époque.

Le deuxième facteur influant sur le choix du style est la compétence disponible pour mener la conception logique. Alors que la technologie actuelle encourage des styles orientés « services », « objets », voire « agents », la culture fonctionnaliste dominante explique la persistance du style fonctionnel dans la réalité des systèmes informatiques, y compris dans les nouveaux développements. Adopter un nouveau style exige de revisiter les compétences de conception et de maîtriser de nouvelles techniques de représentation. L'investissement nécessaire n'est que rarement consenti. Nous tenons là une des explications des difficultés actuelles : on prétend adopter le style à la mode, mais, sans faire évoluer les pratiques et les compétences, on passe systématiquement à côté des concepts nouveaux. Les termes changent, pas les schémas de pensée et encore moins les pratiques. Cela s'appelle : « se payer de mots ».

Le troisième facteur qui conditionne le choix du style est plus subtil : il s'agit des orientations stratégiques. Elles devraient être prises en compte, en majeur, au moment des choix architecturaux. Les transformations significatives des systèmes, impliquant des investissements importants à mener dans la durée, ne peuvent se justifier que pour des raisons stratégiques. Si ce n'est pas le cas, ces transformations ne peuvent pas être portées et légitimées au niveau hiérarchique suffisant. Leurs chances d'aboutir sont alors très réduites.

À titre d'exemple, l'ouverture vers des partenaires et l'extension de la chaîne de valeur entraînent un besoin d'interopérabilité que peut satisfaire une architecture de services. Autre exemple, la motivation économique et le souci de qualité dans un contexte de fusions et acquisitions nourrissent les préoccupations de modularité et de mutualisation des outils. La réactivité ou la capacité à innover, facteurs critiques dans certains secteurs d'activité, imposent souvent de refondre le système informatique ; elles stimulent donc la reconception de l'architecture, avec de fortes exigences sur l'architecture logique.

Le choix du style d'architecture logique ne vaut pas nécessairement pour la totalité du système technique. Quand c'est le cas, les travaux se simplifient, de même que la gestion des compétences. Mais on peut envisager d'appliquer tel style à une portion seulement du système. Le système se compose alors de sous-systèmes de styles différents, soit pour des raisons historiques, soit parce que les caractéristiques des sous-systèmes orientent le choix

du style. Par exemple, dans un système globalement orienté services, on choisit le style événementiel pour un sous-système dont on cherche à réduire le couplage entre les constituants.

Enfin, précisons que la question du style se pose pour chacune des trois facettes. Notamment, le choix d'un style pour la persistance des données est indépendant du choix de style pour le système. Ce point est traité dans les procédés logiques.

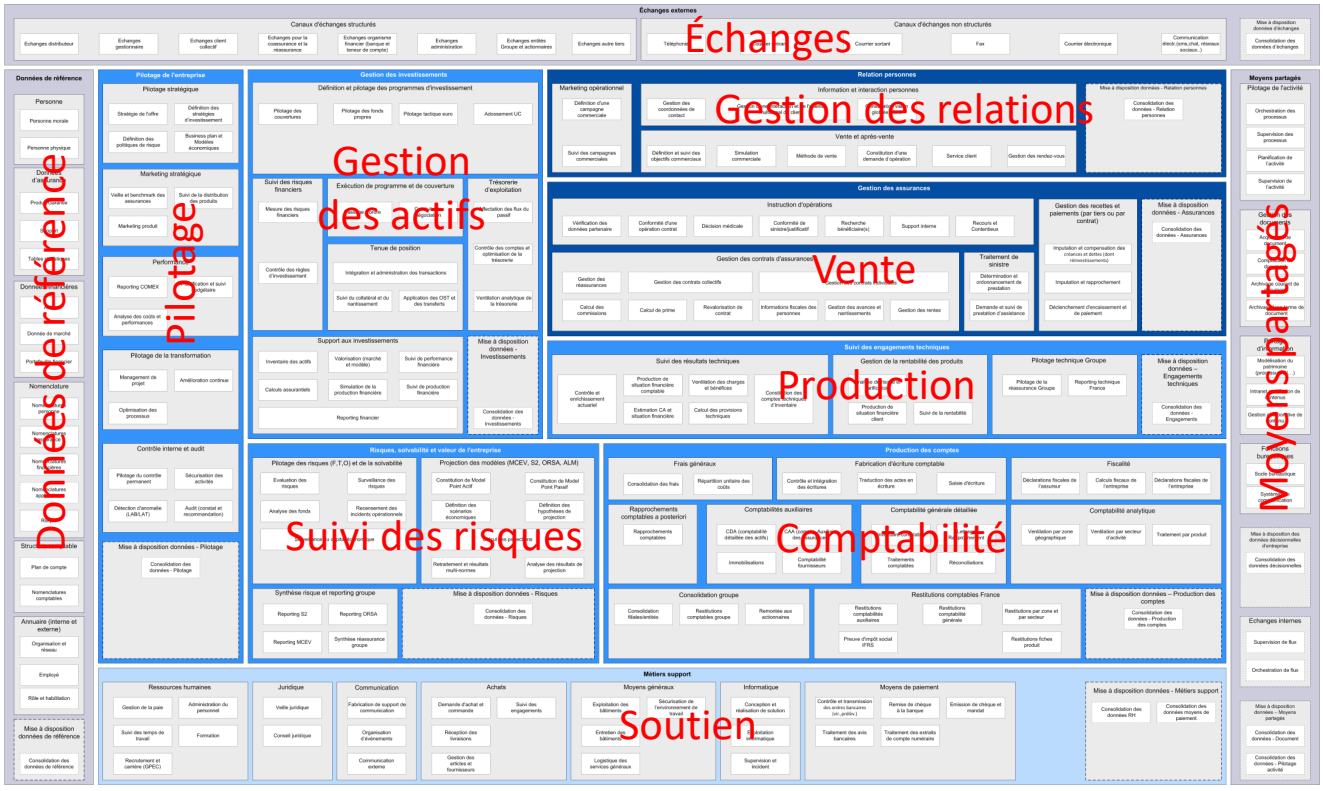
À chaque style sont associées les propriétés attendues du système. En quelque sorte, chaque style porte une promesse. La réalisation effective de cette promesse repose sur la bonne compréhension des exigences associées au style et sur la rigueur de leur mise en œuvre, face à la complexité des systèmes.

c. Architecture fonctionnelle

Le style fonctionnel est le style d'architecture logique dont l'unité constitutive est la fonction. Au sens strict, ce que l'on nomme une architecture fonctionnelle est une architecture logique dans ce style fonctionnaliste[6].

Son critère de structuration est la décomposition hiérarchique descendante de fonctions : à commencer par les grandes fonctions de l'entreprise, jalonnant la chaîne de valeur et reprises sous la forme de « domaines fonctionnels », décomposées en sous-fonctions et ainsi de suite jusqu'à un niveau de grain jugé suffisant.

Figure PxPRD-50_5. Un exemple d'architecture fonctionnelle (architecture logique fonctionnaliste)



Quand on ne se pose pas consciemment la question du style d'architecture logique, par défaut, le style fonctionnaliste s'impose, car il correspond à notre conditionnement culturel :

- D'une part, en pratique, le point de départ de la conception logique a de bonnes chances de se présenter sous la forme d'une liste de verbes, nommant des activités, à travers un cahier des charges, un recensement de « cas d'utilisation » ou un modèle de processus, surtout si l'on fait l'impasse sur la modélisation sémantique.

[6] Le terme « fonctionnel » est devenu ambigu, comme tous les termes dont on abuse. Il évoque le « métier », par opposition à « organique » qui renvoie à la conception informatique (il fut un temps où l'on opposait « analyse fonctionnelle » et « analyse organique » ; à cette opposition s'est substitué le couple analyse/conception). « Fonctionnel » nomme aussi un rôle, celui de l'architecte fonctionnel, lequel peut ou pas adopter le style fonctionnel. Nous préférons le nommer « architecte logique ». Pour parler des composants logiciels, on oppose parfois composants fonctionnels et composants techniques…

- D'autre part, ce paradigme fonctionnaliste s'enracine profondément dans la métaphysique occidentale qui accorde le primat à l'action, au faire, donc la fonction.

On a beau renommer, périodiquement, la fonction avec d'autres termes, tels que : « capacité » (« *capability* »), activité, processus… la matière reste la même. La preuve en est qu'elle se désigne toujours par des verbes d'action.

L'écrasante majorité des systèmes informatiques existants appliquent ce style fonctionnaliste, pour la raison simple que ce paradigme domine notre culture. Quand la question du style architectural n'est pas consciemment posée, c'est le style fonctionnel qui s'impose, exactement comme dans la construction : éluder la question du style condamne au conformisme. Par ailleurs, le mode projet renforce les effets de cette approche, aggravant la redondance et le silotage. Car l'approche fonctionnaliste souffre de plusieurs défauts :

- Son critère de décomposition entraîne la redondance, puisque la structure hiérarchique des fonctions ne laisse aucune place aux éléments partageables et réutilisables.
- L'architecture fonctionnelle commence, au premier niveau de décomposition, avec les « grandes » fonctions, c'est-à-dire les fonctions au sens de la chaîne de valeur, fonctions que l'on retrouve dans l'organigramme de l'entreprise sous la forme d'entités organisationnelles et dans l'architecture des systèmes informatiques sous la forme de domaines fonctionnels (ou blocs fonctionnels) et d'applications. Ces fonctions sont des univers clos, avec leur vocabulaire propre. L'approche fonctionnelle rend aveugle aux objets et notions qui pourraient être mis en commun entre ces silos organisationnels. Partir de cette perception pour construire le système technique aboutit à reconduire ces démarcations et à produire les silos applicatifs. Par exemple, on ne voit pas que derrière la notion de collaborateur, enfermée dans la solution de gestion des ressources humaines, se cache une notion plus universelle, celle de l'individu, qui réapparaît à l'autre bout du système sous d'autres dénominations : prospect, client, partenaire, fournisseur, bénéficiaire, usager, etc.
- L'affaire se révèle plus grave. « Établir un devis » et « Contractualiser » nous arrivent comme deux actions distinctes, avec leur cahier des charges et, parfois, des acteurs et des rôles différents. Ces activités appellent deux projets qui produiront chacun une application. On se retrouve alors avec une table des devis et une table des contrats. Or, en comparant les deux tables, on s'aperçoit que les points communs ne manquent pas. L'approche fonctionnelle part de ces actions et manque les notions, d'autant plus que le vocabulaire professionnel masque facilement les notions les plus universelles. Dans l'exemple, la modélisation sémantique aurait considéré que « devis » est un état d'un contrat, proposé et pas encore signé. L'outillage des deux actions aurait pu s'appuyer sur la même base.
- Proposer une architecture fonctionnelle pour le système technique revient à transposer l'organigramme dans la structure de ce système[7]. Or, deux problèmes surgissent : d'une part, cet organigramme risque fort d'évoluer ; d'autre part, le détail de l'organisation peut varier entre les différentes composantes d'un groupe, par exemple. Ce modèle métier n'est donc ni stable, ni partageable. Est-ce une si bonne idée de le prendre comme point de départ pour la conception du système technique ?

Historiquement, ces silos applicatifs ont été mis en place les uns après les autres, sans plan d'ensemble a priori. Ce n'est qu'a posteriori qu'émerge le besoin de faire circuler l'information entre ces blocs. Et voilà que l'on ajoute des stockages intermédiaires et des tuyaux de toutes sortes pour assurer cette circulation ! Pour rendre justice à la notion de système, nous sommes obligés d'avouer qu'il n'existe quasiment pas de *systèmes* informatiques : en effet, pour que nous puissions parler de systèmes, il eût fallu commencer par en établir les plans, avant de les construire. L'histoire ne s'est pas déroulée de cette façon. Les applications ont été construites, l'une après l'autre, au fil des projets, au gré des préoccupations. Résultat : au lieu de systèmes informatiques, nous avons plutôt, aujourd'hui, des bric-à-brac d'applications.

Quand, consciente des risques qu'apportent son système d'information ou pressée par l'urgence de la transformation, une entreprise se lance dans une refonte, elle n'opte évidemment pas pour le style fonctionnel. Instruite par l'histoire et alertée par l'état des systèmes techniques, elle choisit un autre style et s'empresse de brûler ses schémas d'architecture fonctionnelle.

[7] Ceci rappelle furieusement la loi de Conway.

d. Architecture de services

Le style orienté services ou SOA (*Service Oriented Architecture*) est un style d'architecture logique dont la naissance officielle[8] remonte à 1993. Il s'impose dans la communauté informatique vers le milieu des années 2000. Un jalon important est la publication du « modèle conceptuel » proposé par l'OASIS, en 2006. Ce style a connu un grand engouement et reste dominant. La tendance actuelle autour des micro-services vient réactualiser les principes initiaux – lesquels ont, bien sûr, été perdus de vue au fil des réalisations bâclées et des vulgarisations peu scrupuleuses.

Nous rangeons sous cette appellation, « architecture orientée services » ou « architecture de services », diverses reformulations aux connotations plus ou moins techniques, qui véhiculent toutes le même ensemble de principes, le même paradigme : SOA, REST, micro-services, *API management*.

Ces désignations comportent des nuances techniques importantes (il faut citer aussi les Web Services). Elles visent, en fait, des architectures techniques assez différentes. Mais, quand on en vient aux questions « comment décomposer le système ? », « comment délimiter les composants logiciels ? », « comment trouver les services ? », « quelles interfaces identifier, avec quels mandats ? », alors on se place au niveau logique et les différences techniques n'ont plus lieu d'être.

Bien sûr, la littérature récente, vantant les mérites des nouvelles tendances techniques, s'empresse de dénoncer les versions précédentes. Par exemple, il n'est pas rare de lire, dans les ouvrages sur les micro-services, une critique en règle de SOA. Une fois perpétré ce meurtre du père, il reste à réexprimer les mêmes principes fondateurs. L'exercice peut se révéler salutaire, car ces principes ont souvent été oubliés.

Ces principes, constitutifs du paradigme orienté services, ont été mûris, assemblés, consolidés au fil de l'histoire du génie logiciel. Le tableau suivant cherche à récapituler cette histoire.

Figure PxPRD-50_6. Esquisse d'une histoire des approches d'architecture logique

Ligne du temps (à grosses mailles)	Approche ou style	Apport à l'orientation services
1960	Approche fonctionnelle. Critère : fonction. Décomposition hiérarchique descendante.	Paradigme « naturel », toujours à l'œuvre si l'on n'y prend pas garde. Pertinent dans l'aspect pragmatique (activités métier) ; donc, on le retrouve légitimement dans la strate « Activation »[9].
1970-1980	Approche structurée (analyse-conception structurée). Critère : modularité. Analyse de la qualité structurelle.	Le souci de la qualité structurelle s'exprime à travers des métriques d'architecture. Les préceptes restent actuels : éliminer la redondance, réduire le couplage. Également : recherche de l'autonomie maximale des composants.
1985-1990	Approche client-serveur. Métaphore des relations.	Essentiellement technique, mais la métaphore des relations entre clients et fournisseurs est reprise quasiment telle quelle dans le style orienté services.
1990	Approche orientée objets. Critère : classe. Principe d'encapsulation. Coopération entre les objets.	Cette approche ne se transpose pas au niveau d'un système (du fait de limites techniques), mais SOA retient ce principe fondamental qui suffit à transformer en profondeur les systèmes : l'encapsulation.

[8] Ce sigle apparaît cette année-là dans une conférence du cabinet d'analyse Gartner. De la même année date la création de l'OASIS (*Organization for the Advancement of Structured Information System*).

[9] La notion de strate est présentée au paragraphe 2.3a, p. 17.

Ligne du temps (à grosses mailles)	Approche ou style	Apport à l'orientation services
2000	Approche par composants. Composants et structures. Réutilisation[10].	La préoccupation ne s'arrête pas à la qualité structurelle ; elle vise aussi la réutilisation de composants. Notion de fédération de systèmes. Typologie des composants : processus, entités, utilitaires.

Cette lente maturation explique la robustesse et la persistance de ce style d'architecture sur le marché. Contrairement à l'approche fonctionnaliste, donnée par la culture et réduite à l'intuition non questionnée, l'approche orientée services s'est construite consciemment, en s'appuyant sur un état des lieux des pratiques et sur l'analyse critique des systèmes historiques. Pour la mettre en œuvre, une des difficultés réside dans l'éducation : puisqu'elle n'est pas intuitive, il faut commencer par refaçonner les esprits.

Une autre raison explique la diffusion, voire la popularité, de ce paradigme : de nombreuses offres techniques usent du même vocabulaire et permettent d'incarner les principes dans le système logiciel[11]. La survie d'un paradigme sur le marché dépend de cette facilité à convertir l'expression logique en termes techniques. Inversement, les paradigmes les plus tenaces sont ceux qui émergent comme abstraction de solutions techniques. Dans les faits, on assiste à une élaboration circulaire entre les principes logiques et les solutions techniques. La plus belle démonstration en est l'approche orientée objets, développée à partir d'un socle philosophique et de la volonté de rapprocher de la connaissance humaine. D'où la notion même de classe.

L'architecture de services fait l'objet du guide PxPRD-51. Il détaille les notions propres à ce style et propose une terminologie et un méta-modèle, nécessaires pour asseoir les pratiques.

---

e. Architecture à base d'événements

Dans le style événementiel ou EDA (*Event Driven Architecture*), le mode de communication entre les composants est l'émission et la consommation d'événements. La conséquence est la réduction du couplage à néant : les composants émetteurs et récepteurs n'ont pas besoin de se connaître. Ils peuvent donc évoluer séparément.

La façon de modéliser les constituants logiques et, surtout, leurs relations change complètement. Elle impose une conversion drastique des techniques de modélisation. Le raisonnement diffère complètement par rapport aux approches précédentes.

Des traductions techniques existent : programmation événementielle (déjà courante au niveau des IHM), CEP (*Complex Event Processing*) et MOM (*Message Oriented Middleware* ou Médiaticiel Orienté Messages).

Il est remarquable que la notation BPMN pour la modélisation des processus[12] encourage une approche événementielle. En plus du traditionnel enchaînement entre activités, elle accorde une large place à la notion d'événement et à la circulation des événements. Encore sous-exploitée pour l'instant, elle tend à se diffuser sur le marché. À partir de modèles de processus réalisés dans cet esprit, la conception logique pourra facilement opter pour un style événementiel, au moins pour certaines parties du système.

---

[10] Cf. *Business Component Factory*, Peter Herzum, Oliver Sims, 2000, OMG Press. Cette approche naît de la volonté d'industrialiser la production de logiciel. N'ayant pas trouvé son pendant dans l'offre technique, son empreinte n'a pas marqué le marché.

[11] Ceci se vérifie aussi bien pour la « première génération » de SOA, avec les Web Services et les ESB, que pour la nouvelle génération des micro-services, dotés de compétences techniques (*service mesh*) et associés aux conteneurs et au *Cloud computing*.

[12] Applicable dans l'aspect pragmatique.

f. Architecture orientée aspects

Le terme « aspect », dans la méthodologie Praxeme, est pris dans le sens le plus commun, le plus ordinaire qui soit, sans aucune connotation technicienne. Servant à établir le socle pour résoudre le problème de communication au sein de nos sociétés, la notion doit être elle-même perçue directement.

Il se trouve que ce même terme est utilisé pour caractériser une approche technique, en informatique : la programmation orientée aspects. La POA se situe dans l'aspect logistique (au sens de la Topologie). Cette approche n'est pas opposée à Praxeme, bien au contraire. La méthodologie pourra accueillir, dès le niveau logique, un procédé de conception par les aspects (plutôt que par les services, les fonctions, les événements...). C'est même un prolongement naturel : dès lors que l'on considère que la réalité (entreprise, métier...) présente plusieurs aspects, on se dit légitimement que le système informatique doit en tenir compte. Dans notre approche d'architecture logique, la réponse est la stratification, présentée plus loin (voir § 2.3a, p. 17). Dans une approche orientée aspects, la distribution serait différente. Plutôt que de séparer les aspects à l'échelle du système (les strates), on le ferait à l'échelle des objets, chaque objet présentant les différents aspects. Un aspect, autour d'un objet métier, correspondrait à un certain usage conditionnant la manipulation de cet objet.

On obtient, ainsi, un style d'architecture logique complètement différent. Le modèle MVC s'apparente assez à ce style – en tout cas, il en donne une idée –, à condition de bien noter qu'il se limite à la conception d'une application. Or, quand nous parlons d'architecture, nous nous situons à l'échelle du système. À supposer que les solutions techniques qui s'inscrivent dans ce paradigme se généralisent, il conviendra de vérifier que leur mise en œuvre à l'échelle d'un système entier donne des résultats acceptables, du point de vue de la construction (maîtrise, maintenance) et du point de vue de l'exécution (performances)[13].

g. Pour aller plus loin sur la question des styles logiques

On peut envisager d'autres styles. L'avenir nous apportera, à n'en pas douter, quelques réelles nouveautés. Le méta-modèle applicable à l'aspect logique dépend du style. Il ne sera donc pas traité ici. Le document « Approche orientée services » (référence PxPRD-51s) précise les notions et les règles pour la conception d'une architecture de services, au niveau logique, pouvant se déployer dans plusieurs cibles techniques telles que les Web Services, REST ou les micro-services.

L'orientation services est un style particulier d'architecture logique.

Pour une présentation systématique des styles imaginables en architecture logique, il convient de considérer toutes les possibilités pour assurer 1°) la communication dans le système ; 2°) la coordination. À cela s'ajoute le critère de décomposition. Les modes de communication et de coordination dépendent grandement de la technologie disponible. Les décisions de décomposition sont plus indépendantes ; elles sont plus influencées par des facteurs culturels et organisationnels. Notamment, la manière d'organiser les investissements en informatique, en mode projet, sans plan a priori, détermine la structure résultante, quels que soient la technique ou le vocabulaire adoptés[14].

Le tableau suivant tente une analyse formelle des styles, à partir des modes de communication. Le tableau classe les formes que prend la communication entre les constituants du système, dans l'ordre décroissant du niveau de couplage.

[13] On peut citer AspectJ de Xerox PARC, PHP-AOP, Aspyct (dans le monde Python).

[14] Par exemple, on aura beau parler de « programme API », si l'identification des interfaces se fait à l'intérieur des projets, il n'y a aucune chance d'améliorer la structure du système. La « loi de Conway » se vérifie parfaitement dans les systèmes historiques : la structure d'un système reflète celle de l'organisation qui l'a produit. La plupart du temps, elle tombe assez loin de la structure optimale. Cette « loi » neutralise toute volonté de transformation en profondeur.

Figure PxPRD-50_7. Détermination des styles par les modes de communication

Forme de la communication		Description	Conséquence sur le couplage
Appel	**Procédure**	Pas de paramètre. L'information est partagée dans une zone commune (*data division*, *data pot*).	Fort couplage (appel, GOTO…). Pas de protection des données.
	Fonction	Passage des informations par les paramètres et le résultat.	Meilleure protection des données. Style fonctionnel amélioré.
	Opération	Fonction associée à un constituant responsable – exclusif – des données.	Encapsulation (protection de données). ➔ orientations objets et services[15].
Pas d' appel	**Message**	Émission de l'information vers un destinataire connu.	Couplage moindre : les constituants se connaissant, mais pas au niveau des opérations (pas d'appel).
	Événement	Émission de l'information. Pas de destinataire connu.	Aucun couplage : les constituants s'ignorent mutuellement. Ils sont à l'écoute des événements.

Quel que soit le moyen de communication retenu dans le système, reste la question de la coordination entre les constituants, au moment de l'exécution. Le tableau suivant présente les options envisageables pour cette coordination.

Figure PxPRD-50_8. Les grandes options pour la coordination de l'exécution au sein du système

Moyen de coordination	Description	Conséquences
Appel explicite	Les appels sont programmés « en dur ». Typique des générations « procédure » et « fonction », mais se retrouve aussi dans les opérations et les services.	Crée des dépendances fortes, au niveau le plus détaillé, c'est-à-dire dans les algorithmes.
Appel médiatisé	Les appels de l'intérieur des constituants ne visent pas directement les constituants fournisseurs.	Introduction d'interfaces « requises », de *proxies*… ➔ meilleure maîtrise du couplage.
Orchestration	Coordination par le truchement d'un contrôleur central (par exemple, un moteur d'exécution des processus qui se charge de l'enchaînement des activités et de l'invocation des services).	L'enchaînement peut alors évoluer facilement, sans impact sur le logiciel. ➔ agilité (au moins sur ce qui est pris en charge par le contrôleur).
Chorégraphie	En l'absence de contrôleur central, les constituants se coordonnent de proche en proche, par émission de messages ou d'événement.	On gagne sur les deux tableaux : réduction du couplage ; évolutivité. Tout repose sur la conception des événements. L'approche change totalement.

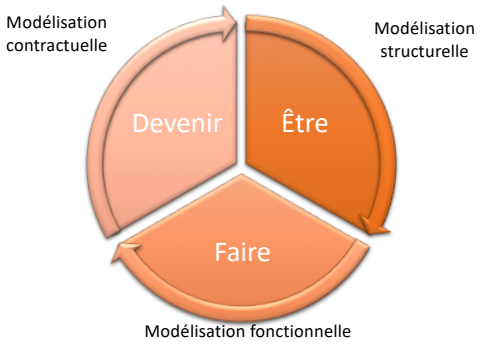
2.3 Catégories de représentation

Les catégories de représentation sont les notions opératoires dont nous nous servons pour nous représenter le système, ici dans son aspect logique. Ce paragraphe s'arrête aux catégories de représentation qui sont valables

[15] Dans l'orientation objets, le constituant auquel sont associées les opérations est la « classe », celle-ci représentant – normalement – un ensemble cohérent d'objets métier ou un concept. Dans l'orientation services, le constituant reste à définir : c'est tout l'objet de la méthode Praxeme pour SOA, introduite dans PxPRD-51s.

pour tous les styles d'architecture logique. Les catégories propres à un style particulier, comme le « service », sont présentées dans le guide dédié à ce style.

Figure PxPRD-50_9. Approche de modélisation par triangulation

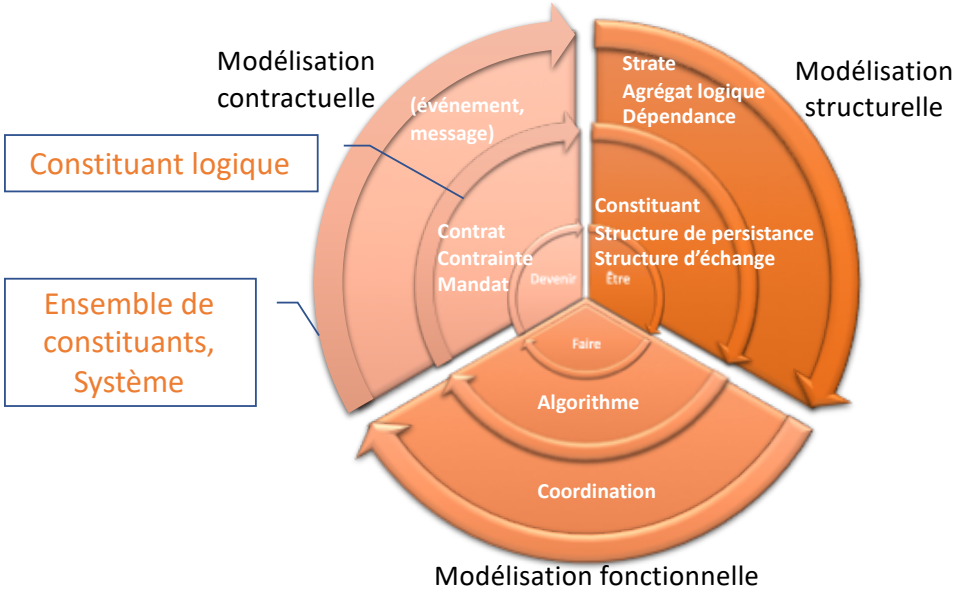


Comme pour tous les autres aspects substantiels, la description logique d'un système ou d'un de ses constituants ne saurait être complète que si elle couvre les trois axes :

- Axe structurel : de quoi le système est-il fait ? (l'être, la substance) ;
- Axe fonctionnel : que fait-il ? comment le fait-il ? (le faire, la fonction, le fonctionnement) ;
- Axe contractuel : comment se comporte-t-il ? quelles sont les contraintes à respecter ? (le devenir, les contraintes, la transformation).

Selon les styles, les termes pour exprimer la matière dans ces trois axes sont plus ou moins naturels. La figure suivante en indique quelques-uns, sans évoquer un style quelconque. Ils sont introduits dans la suite de ce chapitre.

Figure PxPRD-50_10. Les termes généraux de l'aspect logique



a. Strates

Le terme « strate » est équivalent à celui de « couche ». Simplement, parce que l'architecture technique définit des couches, en nombre souvent plus important et selon des besoins qui lui sont propres, il a été jugé préférable d'utiliser un autre terme, dans l'univers logique, ceci afin d'éviter les confusions.

L'identification des strates est indépendante du choix de style. La méthode les fixe, une fois pour toutes, en appuyant leur définition sur un raisonnement théorique, difficilement contestable.

Le point de départ de ce raisonnement se trouve dans le cadre de représentation, la Topologie du Système Entreprise. Ce cadre distingue l'aspect sémantique (niveau conceptuel) et l'aspect pragmatique (niveau organisationnel). Cette distinction est classique, et obéit au principe de séparation des niveaux d'abstraction. Elle permet d'extraire de la représentation du métier, ce qui en forme le noyau stable et partageable de connaissances : les fondamentaux du métier.

Le bon sens veut qu'en construisant le système technique, on maintienne cette séparation, et que l'on isole des constituants qui abritent cette connaissance fondamentale. Les concepts qui l'expriment se traduisent en constituants logiques, sans aucune adhérence avec des éléments liés à l'organisation. De cette façon, ces

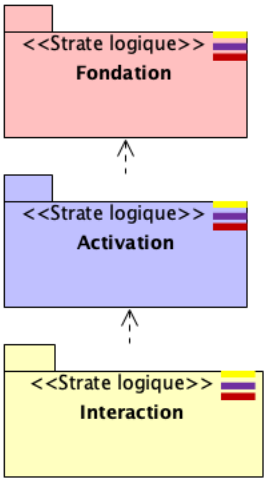
constituants jouiront de la même stabilité que les concepts dont ils dérivent. Certains seront autant réutilisables que les concepts dont ils dérivent sont universels.

Ainsi, l'aspect sémantique donne lieu à une partie bien délimitée du système : la strate « Fondation ». Cette strate sera peuplée uniquement des résultats de la dérivation appliquée aux éléments sémantiques.

L'aspect pragmatique fournit un autre point de départ pour la conception logique. Évidemment, les constituants déduits des éléments pragmatiques ne doivent, en aucun cas, se retrouver mélangés avec les constituants précédents. Ils seront rangés dans une autre strate : la strate « Activation ».

Une troisième strate est nécessaire, pour loger des constituants qui prennent en charge les interactions du système avec l'extérieur – son environnement. Les composants d'interaction couvrent les interfaces homme-machine, ainsi que le couplage avec d'autres systèmes : ils vérifient les conditions d'interaction (habilitation, protocole, langage…) et pilotent le dialogue. Les constituants logiques de cette strate aident à concevoir les interactions, pour un style d'échange particulier : par exemple, interface de type smartphone, opposée à une IHM sur ordinateur de bureau ou à un échange avec un système partenaire. Dans ces trois cas, les modalités du dialogue diffèrent, ce qui peut conduire à une conception distincte. Plutôt que de réaliser directement l'IHM, on peut trouver un intérêt à établir une spécification logique, qui vaudra pour plusieurs cibles techniques (iOS et Android, par exemple).

Figure PxPRD-50_11. La stratification de l'aspect logique (à la mode platonicienne : le plus abstrait au sommet)



La stratification

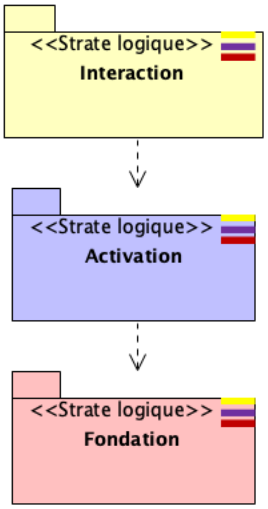
Au premier niveau de décomposition de l'aspect logique, l'architecture distingue :
- la strate « Fondation », reprenant le contenu de l'aspect sémantique ;
- la strate « Activation », correspondant à l'aspect pragmatique ;
- la strate « Interaction », traitant des interactions du système avec son environnement.

Ce principe n'est pas une décision d'architecture : il s'impose, dès lors que l'on adopte le cadre de représentation. Il tire les conséquences de la détermination de l'aspect logique par les deux aspects amont. Il conduit à un saut qualitatif : si l'on s'y tient, on reconstruit le système technique en isolant un noyau stable et largement partageable. La physionomie du système résultant change complètement ; le taux de réutilisation augmente, et l'interopérabilité progresse.

Les désignations des strates peuvent changer[16], pas leur nombre ni leur justification. On pourrait être tenté d'ajouter une strate « Agilité » ou « Sécurité » ou « Pilotage »… C'est un réflexe pavlovien : dès que l'on veut montrer que l'architecture prend en compte une préoccupation, on ajoute une case sur le premier schéma ! Rappelons que les thèmes cités en exemple reçoivent un traitement précis, bien plus efficace qu'en les isolant dans une zone du système. Le graphe d'architecture logique ne doit pas être détourné pour des motifs de communication.

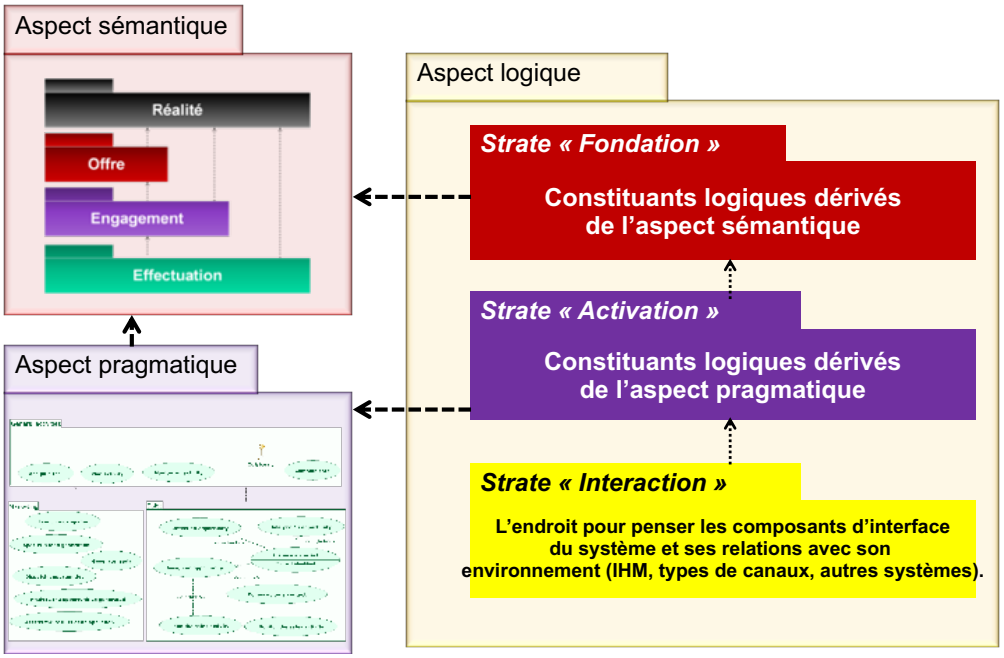
[16] Changer les termes d'une méthode conduit à renoncer à sa fonction d'homogénéisation du vocabulaire, et à s'exposer à quelques dépenses dans la communication avec d'autres interlocuteurs. « Fondation » évoque les fondamentaux du métier ; également, l'idée que c'est par là qu'il faudrait commencer la construction du système. « Activation » comme « activités métier ». « Interaction » pour rappeler que le système n'est pas fermé, qu'il s'ouvre sur un extérieur et que les interactions avec son environnement constituent un sujet à ne pas négliger.

Figure PxPRD-50_12. La stratification de l'aspect logique (façon BTP : les fondations en-dessous)



En pratique, la définition des strates par la méthode permet, à l'architecte, de gagner du temps. Au moment de la mise en place du Référentiel de Description de l'Entreprise, la base de modélisation est déjà structurée en aspects. Quand on aborde l'aspect logique, on découvre, au premier niveau, les trois strates. Le travail d'architecture commence ensuite.

Figure PxPRD-50_13. La justification de la stratification logique



b. Agrégats

Parmi les constituants logiques, les plus visibles sont les agrégats ou blocs[17] : ensembles cohérents d'éléments logiques. Tout l'art de l'architecture logique consiste à décomposer le système en agrégats, sans redondance et avec le minimum de couplage.

Le terme « agrégat » est, volontairement, très général. Dans certains contextes, on peut lui préférer « système » et « sous-système ». Dans d'autres, « application ».

De la figure précédente, on peut déjà subodorer qu'un certain niveau d'agrégats se déduira facilement de la structure des aspects antécédents.

Dans la tradition méthodologique, l'architecture logique se déploie sur deux ou trois niveaux d'agrégation[18]. Sans qu'il n'y ait jamais eu de justification théorique, l'expérience tend à montrer qu'avec trois niveaux d'agrégation, on arrive à maîtriser les systèmes artificiels. Cette justification empirique suffit à fonder une méthode.

Pour aller plus loin et nommer les agrégats aux différents niveaux de décomposition, il faut se donner un vocabulaire. Celui-ci, nécessairement, s'appuie sur une métaphore : c'est une des caractéristiques de l'aspect logique ; n'étant plus la réalité « métier », pas encore celle de la technique, il lui faut un vocabulaire, si possible imagé, pour nommer ses constituants. Pour bien faire, la métaphore devrait être choisie en cohérence avec le style

[17] Nous prenons soin, ici, d'éviter des termes qui trahiraient le choix d'un style.

[18] Méthode TACT : machine logique et serveur logique. Urbanisation du SI : zone, quartier, îlot. Peter Herzum : fédération, systèmes, composant…

retenu. Le guide PxPRD-51s, « Approche orientée services », se prête à l'exercice, et propose un vocabulaire que l'architecture logique peut appliquer, dès lors qu'elle opte pour ce style.

c. Interfaces

On attache beaucoup d'importance aux agrégats, parce qu'ils apparaissent comme les morceaux, presque tangibles, du système. Toutefois, nous devons accorder encore plus d'importance aux interfaces, lesquelles vont conditionner la qualité de l'architecture. Certes, découper un système en sous-systèmes conduit à faire apparaître des interfaces. L'acte semble unique : la décision de découper crée, du même coup, deux morceaux et une interface. À cela, il y a deux objections : l'une culturelle, l'autre logique.

- Première objection : quand on examine les représentations des systèmes informatiques, sous forme de POS[19] ou de diapositives (autant dire, sans formalisme), force est de constater que les interfaces ne sont pas décrites, ni même les dépendances entre les blocs représentés.
- Deuxième objection : fonctionnellement, un système est plus déterminé par ses interfaces que par ses agrégats. En effet, on peut imaginer qu'un même ensemble d'interfaces soit servi par une structure en agrégats ou une autre. Pourvu que les interfaces soient les mêmes, le système fonctionnera de la même façon et rendra les mêmes services.

Cette observation peut paraître contre-intuitive. En tout cas, elle choque notre conditionnement substantialiste. Elle aura des conséquences sur la manière d'organiser le travail, comme on le verra dans le chapitre 4, « Utilisation des modèles logiques ».

Ce qui se passe au niveau des interfaces est plus important que le découpage en constituants.

Nous ne disons rien, pour l'instant, de la manière de spécifier, puis de réaliser une interface[20]. Une interface est un point de contact entre un agrégat et un autre. Le terme « interface » peut recouvrir tout un spectre de notions logiques, à un extrême : la simple dépendance permettant un passage ; à un autre : un composant en charge des communications entre les morceaux du système.

Les interfaces peuvent aussi se trouver *à la périphérie* du système, pour y assurer le contrôle des interactions.

d. Facettes

Dans l'aspect logique apparaît une partition qui n'avait pas lieu d'être dans les aspects antécédents. La description logique comprend trois facettes : les constituants, les données et les échanges.

De façon plus générale, décrire un système implique de documenter :

- sa substance, sa composition (les agrégats et leurs relations) ;
- la persistance de l'information ;
- les échanges au sein du système et avec son environnement.

Chacune de ces facettes assemble des éléments d'une nature particulière, respectivement : les agrégats logiques ; les structures de persistance ; les structures d'échange. Chaque facette obéit à ses propres règles de construction. Les choix techniques pour les convertir en logiciel sont indépendants.

Mais, contrairement aux strates et aux agrégats, les facettes ne donnent pas la structure du système. Il ne faut pas s'imaginer qu'en ouvrant le système, on découvre trois facettes comme des morceaux distincts. Elles sont une

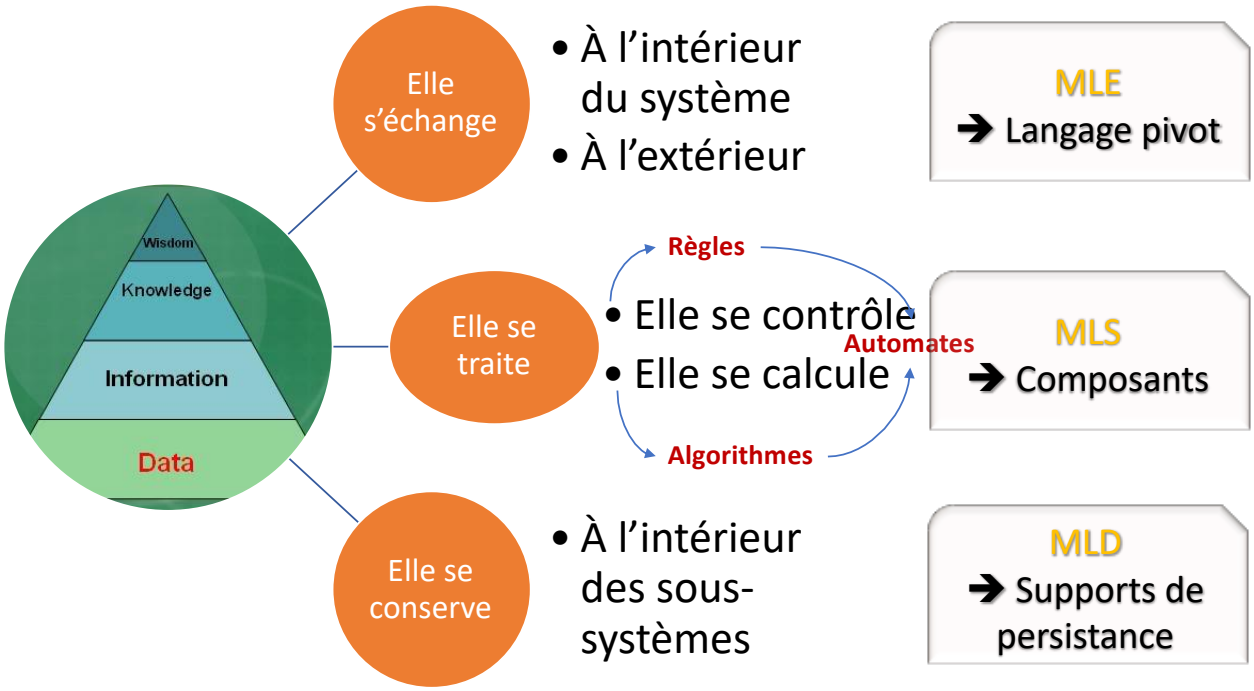
---

[19] POS : Plan d'Occupation des Sols. Représentation d'un système informatique, de niveau logique, dans le courant de l'urbanisation de système d'information.

[20] Même en informatique, l'interface n'est pas nécessairement une API.

façon de rassembler des éléments de même nature, dans une perspective particulière. Elles se manifestent à travers trois types de modèles logiques. On s'attend à trouver : des modèles logiques de données, des modèles logiques d'échange, des modèles logiques de constituants. Parmi les décisions d'architecture logique, il faudra préciser leur articulation. La réponse repose sur le choix du style et le niveau d'audace souhaité pour améliorer le système[21].

Figure PxPRD-50_14. Les facettes de l'aspect logique



MLE : modèle logique des échanges, spécification du langage pivot (sa traduction technique).

MLS : modèle logique des services ou, plus généralement, des composants.

MLD : modèle logique des données.

Le système artificiel s'appréhende à travers trois facettes :
- sa composition (et sa cohabitation avec d'autres systèmes) ;
- sa communication (interne et externe) ;
- son information (la réalité de son état interne).

2.4 Principes directeurs de l'approche logique

Un principe directeur est une règle générale qui guide la conduite. Dans la pratique professionnelle, il inspire les décisions et intervient dans leur justification. Pour faciliter leur application et leur transmission, nous devons nous limiter à un petit nombre de principes. Quand une organisation affiche de nombreux « principes », elle confond la notion de principe avec celle de décision d'architecture. Pour éviter ce travers, revenons à l'étymologie du terme « principe » : origine, cause première. De fait, un principe n'a pas besoin d'être justifié ; en toute logique, il ne saurait l'être puisqu'il est au commencement de tout raisonnement. Tout le monde y adhère ou est censé y adhérer. Dans la réflexion, il fonctionne comme un axiome que l'on pose au début d'une théorie, sans le justifier[22].

[21] La question sera traitée dans le procédé PxPCD-52, « Architecturer l'aspect logique du système ».

[22] Faire passer une décision ou un choix pour un principe, c'est donc l'exclure de la discussion, en un coup de force finalement peu démocratique. Cela revient donc à brider l'exercice de la rationalité.

En méthodologie, un principe est une idée de bon sens, admise par la majorité des praticiens dans le domaine de pratique ou la discipline où il s'applique[23]. Formuler un principe est une façon de résumer les bonnes pratiques et de les communiquer. Nous avons besoin de ces repères – stables – pour contrer le mouvement permanent des modes et l'évaporation des connaissances qu'il entraîne. Derrière le renouvellement des termes et les effets d'annonce, on ne trouve qu'un petit nombre de principes, que les générations successives se transmettent. Ils doivent inspirer les décisions d'architecture et de conception. Au premier abord, ils peuvent paraître abstraits, mais l'apprenti architecte a intérêt à les assimiler le plus tôt possible pour accéder à la maîtrise de l'art.

Déjà, aux fondements de la méthodologie, nous trouvons un principe absolu, celui de la séparation des niveaux de préoccupation (*separation of concerns*). Sur la base de ce principe, l'aspect logique a été isolé (voir l'introduction). Ensuite, nous pouvons formuler une poignée de principes propres à l'aspect logique :

1. l'indépendance logique-technique ;
2. l'alignement de la conception logique sur les aspects amont ;
3. les principes contribuant à la qualité structurelle ;
4. certains principes plus précis, liés à un paradigme ou à un style.

Ces principes, détaillés ci-après, guident la prise de décision en architecture logique.

a. Indépendance logique-technique

Conséquence de la définition de l'aspect logique comme abstraction du système technique, le modèle logique se veut indépendant des choix techniques. Sa finalité est de décrire le système à construire, dans son aspect logistique, avec ses moyens matériels et logiciels, mais de façon suffisamment abstraite pour se garantir de la diversité et des évolutions technologiques. La règle est toujours de sérier les décisions : au niveau logique, il s'agit de mettre au point la structure optimale du système ; au niveau logistique, le travail consistera à convertir la spécification logique dans les termes d'une cible technique.

Régulièrement, l'architecte logique devra mettre en avant ce principe d'indépendance, pour préserver la conception logique des a priori d'une solution technique. Le danger est grand, en effet, de brider la conception et de la lier à une cible technique particulière. Cette attitude conduirait à :

- limiter la capacité d'invention, lors de la structuration et de la conception du système[24] ;
- rendre le modèle logique moins communicable, puisque compliqué par une terminologie technique ;
- faire dépendre la conception logique des choix techniques, donc limiter la durée de vie et l'utilisation de la spécification logique qui ne pourrait plus, alors, accompagner la transformation du système sur le long terme.

Ce principe d'indépendance logique-technique présente un intérêt défensif : les architectes et concepteurs logiques y recourent pour défendre leur discipline et leur périmètre de responsabilité, mis à mal par les pratiques actuelles.

Toutefois, l'application de ce principe se heurte à une limite : il est indispensable d'assurer la convertibilité de l'expression logique dans les termes techniques. Cette exigence peut peser sur les moyens d'expression utilisés dans l'aspect logique. Elle appelle un exercice particulier, prévu par la méthode et connu comme la « négociation logique-technique ». Le chapitre 4 détaille ce moment essentiel dans la transformation des systèmes.

[23] Nous convenons que cette notion de « bon sens » est suspecte. D'ailleurs, ce qui est considéré comme évident par une génération peut se trouver contesté et subverti par la suivante, au moment d'un changement de paradigme. La décomposition fonctionnelle en fournit un bel exemple : tenue pour évidente pendant des décennies, elle est mise en cause par la logique objet et son principe d'encapsulation, repris dans le style orienté services (cf. PxPRD-51s).

[24] Dans les travaux d'architecture fonctionnelle ou d'urbanisation du SI, on observe souvent que la considération du système existant inhibe l'élaboration de l'architecture.

Le principe d'indépendance logique-technique garantit la lisibilité et la stabilité de l'expression logique, sur le long terme.

b. Alignement

L'alignement des solutions techniques sur les besoins de l'entreprise et sur ses orientations stratégiques s'impose comme une revendication légitime. En termes de méthode, cette revendication revient à soulever la question du point d'origine pour la conception logique. Praxeme y répond en révélant les deux dépendances de l'aspect logique : à l'égard de l'aspect sémantique, d'une part ; à l'égard de l'aspect pragmatique, d'autre part. Pour ces deux chemins, mis en place dans la Topologie du Système Entreprise, la méthode propose des règles de dérivation. À partir des éléments des aspects antécédents, ces règles produisent mécaniquement des éléments de l'aspect logique. Une grosse partie des constituants logiques, de l'ordre de 80% de la substance du système, émerge de cette façon.

Le principe de dérivation garantit l'alignement de la solution sur la réalité du métier, en aidant à produire la solution en référence à cette réalité.

Le mécanisme de la dérivation s'appuie sur une technique décrite dans le standard *Model Driven Architecture* (MDA) de l'OMG.

Il n'est possible d'établir les règles de dérivation que si l'on connaît la nature des éléments de destination. Ces règles de dérivation ne valent, donc, que pour un style donné. C'est pourquoi ce guide n'en dira pas plus. Elles ne peuvent être exposées que dans les documents liés à un style logique.

Associée à ce mécanisme de la dérivation, la traçabilité s'installe comme une conséquence naturelle : tout élément obtenu par dérivation reste lié à son élément d'origine. De cette façon, en cas de changement dans le modèle amont, l'analyse d'impact est menée rapidement et sans faille, pour déterminer les implications dans les aspects en aval.

Traçabilité : capacité à reconstituer une chaîne de détermination.

Cette capacité repose sur un minimum de discipline : les modélisateurs doivent s'imposer de mettre en place les liens de traçabilité, à chaque fois qu'ils créent ou documentent un élément qui dépend d'un autre. Si cette précaution est prise au moment de la conception, l'analyse d'impact en cas d'évolution pourra être menée quasi instantanément, économisant alors une importante charge de travail et évitant le risque d'un oubli.

c. Qualité structurelle

Les méthodes d'analyse-conception structurées ont jeté les bases pour une évaluation rigoureuse de la qualité d'un système informatique. Leurs métriques se transposent aisément à tout type de système technique.

La terminologie établie à cette époque (années 60-70) reste éminemment pertinente : redondance, couplage, autonomie, dépendance… L'intérêt pour la qualité structurelle des systèmes naît des préoccupations de maîtrise, d'évolutivité et d'économie. Des systèmes lourdement redondants – comme ceux que l'on observe presque partout – sont coûteux, à la fois, à maintenir et à faire fonctionner. Ils entravent les évolutions.

Pour apprécier la qualité structurelle des systèmes, plusieurs métriques sont nécessaires. Citons : taux de redondance, taux de couplage, taux de réutilisation, le ratio entre le nombre d'interfaces et le nombre d'agrégats. Pour guider les décisions d'architecture, deux principes soutiennent le raisonnement :

1. éliminer la redondance ;

2. réduire le couplage.

Les paragraphes suivants détaillent ces deux principes.

Figure PxPRD-50_15. Un point sur le vocabulaire de la qualité structurelle

Terme	Définition	Commentaire
Redondance	Fait d'apporter plusieurs réponses à un même besoin	Elle peut être cachée, car les concepts sont déguisés sous les mots.
Couplage	Existence d'une dépendance entre deux éléments, nécessaire au fonctionnement d'au moins un d'entre eux	Impossible à supprimer totalement : un moteur sans courroie de transmission ne nous emmènerait pas loin
Autonomie	Absence de dépendance	Une vertu, à tous points de vue[25]
Modularité	Qualité d'un système construit à partir de modules bien identifiés	Un « bon » module est un élément de ce système obéissant à des exigences telles que l'autonomie et la réutilisation : les modules sont, souvent, partagés.

d. Élimination de la redondance

La non-redondance recherchée s'interprète au niveau logique. Un élément de modélisation « métier » ne doit avoir qu'une et une seule traduction dans l'aspect logique, répercutée tout de même à travers les trois facettes. Dans le cas où la modélisation du métier souffre elle-même de redondance, il faudra un effort supplémentaire pour l'éliminer de l'aspect logique. Le plus simple est de porter l'exigence de non-redondance aussi sur les aspects sémantique et pragmatique, avant d'appliquer les règles de dérivation. Par exemple, la notion universelle d'individu a pu être masquée par un vocabulaire mettant en avant les rôles : client, prospect, collaborateur, fournisseur, usager, etc. Reprendre ces rôles sous la forme d'autant de constituants logiques se révélerait catastrophique[26]. Dans l'aspect logique, on doit pouvoir repérer un constituant unique en charge de la notion d'individu, quelle que soit la situation dans laquelle elle se manifeste.

Pour autant, l'unicité du constituant logique ne se retrouve pas forcément ensuite. À un même constituant logique peuvent correspondre plusieurs composants logiciels. Le cas se produit quand une même conception logique se projette dans plusieurs architectures techniques. Le principe de non-redondance que nous posons ici vaut pour l'aspect logique seul.

Il peut arriver aussi que, pour des raisons réglementaires ou des soucis de performance, l'architecture physique instancie plusieurs fois le même composant. Une telle décision, bien sûr documentée et administrée, se légitime parfaitement. Cette forme de redondance, dans l'aspect physique, ne contredit en rien l'idéal de non-redondance qui inspire l'architecture logique.

Le principe de la non-redondance vise à l'économie du système, en réduisant le volume de sa spécification par élimination de constituants inutiles.

Cette recherche de la non-redondance incite à identifier des constituants mutualisables. Elle prépare la réutilisation. Le taux de réutilisation des composants d'un système augmente en même temps que son taux de redondance diminue. Dans ce combat contre la redondance, l'architecture doit embrasser les trois facettes de

[25] Particulièrement chez les misanthropes.

[26] C'est pourtant exactement ce qui s'est passé depuis un demi-siècle et qui continue de se produire, faute de méthode de référence et d'un management volontariste pour changer les choses.

l'aspect logique. On comprend qu'une architecture des données redondante fait craindre la redondance des constituants qui gèrent ces données.

e. Réduction du couplage

En éliminant la redondance, l'architecture logique fait apparaître des constituants réutilisables. Par construction, ceux-ci sont sollicités par d'autres constituants, ce qui augmente le couplage. Or, un couplage exagéré rend le système difficile à maîtriser. Intervient, alors, le deuxième principe fort de l'architecture logique : la limitation du couplage.

Le principe de limitation du couplage : l'architecture logique recherche le niveau de couplage strictement nécessaire au fonctionnement du système.

Le cas idéal, pour un constituant, est l'autonomie : un constituant totalement autonome, ne dépendant d'aucun autre constituant, ne subira pas les évolutions d'autres constituants ; traduit en termes techniques, il donnera lieu à un composant facile à déployer.

Il n'est pas rare qu'un agrégat logique obtenu par dérivation d'un élément amont arrive avec plusieurs dépendances. L'architecte logique cherchera, alors, à éliminer tout ou partie de ces dépendances, afin d'augmenter l'autonomie de cet agrégat. Conséquemment, il réduit le couplage au sein du système.

Un taux de couplage élevé, dans un système, révèle presque toujours la faiblesse de la réflexion architecturale. Le cas se produit dans la construction des systèmes informatiques : sans plan d'ensemble *a priori*, les applications ont été développées les unes après les autres ; la nécessité de faire circuler l'information entre elles est apparue *a posteriori* et a donné lieu à l'ajout de dispositifs ad hoc. Cette histoire non planifiée explique, en partie, la complication de ces systèmes.

S'intéresser au couplage conduit à examiner comment les constituants sont reliés entre eux. Cet examen ne s'arrête pas au bilan des dépendances – les liens ; il porte aussi sur la nature de ces liens. Que se passe-t-il à l'interface ? Cette notion d'interface a fini par être réifiée, c'est-à-dire que les architectures contiennent aujourd'hui des composants que l'on reconnaît comme interfaces. Ils obéissent à des règles et protocoles ; ils assurent la communication entre les composants. Cette notion réifiée d'interface joue un rôle cardinal dans les architectures de services.

La notion d'interface, surtout sous sa forme réifiée, est un outil déterminant dans les mains de l'architecte logique.

Ce point deviendra évident dans le procédé d'architecture logique[27].

f. Antagonisme des deux principes

Les deux principes, élimination de la redondance et réduction du couplage, forment le bréviaire de l'architecture logique. À eux deux, ils suffisent à éclairer les décisions prises en élaborant la structure optimale du système. Les autres propriétés recherchées du système découlent de l'action de ces deux principes : la modularité, la réutilisabilité, l'économie (la limitation du volume, notamment), l'autonomie, la maintenabilité. L'agilité du système, c'est-à-dire sa capacité à s'adapter rapidement et à moindre coût, dépend en grande partie de la qualité structurelle. Elle implique, également, de mettre en place des dispositifs d'agilité et d'en tirer certaines conséquences. Mais ces dispositifs, déployés dans un système mal structuré, perdraient beaucoup de leur efficace.

Si ces deux principes sous-tendent l'essentiel du raisonnement architectural, il importe de bien se pénétrer de ce constat :

---

[27] Cf. PxPCD-52, « Architecturer l'aspect logique du système ».

Les 2 principes de l'architecture logique sont antagoniques.

En effet, pour atteindre l'autonomie d'un composant, il n'y a rien de plus simple que d'y injecter des éléments pris ailleurs. Cette manœuvre sera fondée, au niveau technique ; elle correspond au mécanisme de l'importation. Mais elle n'est pas admise au niveau logique. La description logique doit prévoir cette relation d'importation, justement pour éviter la duplication (voir le chapitre 3). Donc, éliminer la redondance conduit à isoler des constituants et à les lier à d'autres, donc à augmenter le couplage. Le dilemme se pose régulièrement. Et l'architecte d'arbitrer entre plusieurs solutions possibles. C'est précisément dans cette dialectique que réside l'art de l'architecte logique. Ce travail de recherche du meilleur équilibre ne peut s'exercer de façon pertinente qu'à une échelle suffisante : celle du système ou de la fédération de systèmes. Il est sans intérêt au niveau des projets.

La qualité structurelle d'une architecture logique s'améliore quand on élimine la redondance et que l'on réduit le couplage. Ce faisant, on augmente l'autonomie des composants et la réutilisation. Cet effort prépare la qualité du système technique.

Pour l'architecte logique, l'application de ces deux principes est une obsession, presque une raison d'être !

2.5 Détermination de l'aspect logique

La nature et les principes de l'architecture logique étant posés, revenons sur la question de son contenu. Les questions suivantes préparent les pratiques :

- Par quoi, donc, l'aspect logique est-il déterminé ?
- D'où provient son contenu ?
- Quelles influences s'exercent sur lui ?

Nous nous arrêtons ici aux lignes de force qui déterminent le contenu de la conception logique, les pratiques détaillées dépendant toujours du style logique.

a. Détermination par le métier : l'alignement sur les besoins et orientations de l'entreprise

Conformément au principe d'alignement (§ 2.4b) et à la position de l'aspect logique dans le cadre de représentation (§ 1.2), la conception logique s'alimente du contenu des aspects

1. sémantique (« objets métier ») ;
2. pragmatique (activités « métier »).

Le travail est facilité par l'application de règles de dérivation, décrites dans les procédés (voir le § 5.2 qui présente les procédés logiques).

b. Détermination par l'aspect intentionnel : la prise en compte des préoccupations

La Topologie du Système Entreprise prévoit, également, une dépendance de l'aspect logique vers l'aspect intentionnel. D'ailleurs, tous les aspects dits « substantiels » dépendent de l'aspect intentionnel. Cette dépendance autorise les références à partir des éléments logiques vers :

- les valeurs de l'entreprise ;
- les objectifs et les exigences ;
- les métriques (indicateurs) ;
- le vocabulaire.

Le passage de l'aspect logique vers l'aspect intentionnel ne permet pas de mettre en place des règles de dérivation. Celles-ci ne se jouent qu'entre les autres aspects. À partir de l'aspect intentionnel, seule se met en place la

« projection ». Elle se traduit par un lien de traçabilité, de l'élément logique vers l'élément d'intention. Cette relation intervient dans la justification de certaines décisions d'architecture. Elle permet de montrer comment une préoccupation a été prise en compte dans la solution, dans le cas où la réponse a pu être apportée à travers l'aspect logique.

Quelques exemples :

- La préoccupation concernant les relations avec les clients, préoccupation qui peut elle-même provenir du système de valeurs ou des objectifs[28], réclame la capacité de produire rapidement, à l'exécution, la « synthèse client ». Une architecture dans laquelle l'information relative au client est pulvérisée à travers tout le système aura du mal à satisfaire cette attente. Un référentiel des personnes, un dispositif pour la qualification des objets, un constituant centralisant les interactions du client avec la totalité de l'organisation… constituent des éléments de réponse qui peuvent apparaître sur un graphe d'architecture logique.
- L'analyse stratégique peut révéler un besoin d'agilité sur une partie des activités ou des concepts. Par exemple, la demande d'agilité peut concerner certaines activités de la chaîne de valeur, quand on anticipe des reconfigurations organisationnelles : changement de l'organigramme, évolution des compétences, fusions-acquisitions… Autre exemple, concernant cette fois-ci les fondamentaux du métier, la stratégie réclame plus de souplesse dans la définition des offres (*bundle*, *pricing*, assimilation des catalogues des filiales, etc.). Ces deux exemples peuvent évoquer deux types de dispositifs d'agilité proposés par le marché, respectivement : le moteur d'exécution de processus et le moteur de règles. En tous les cas, ces éléments d'intention conditionnent des choix d'architecture logique et peuvent modifier le détail de l'expression logique.

L'architecte logique, autant que l'architecte métier, explore l'aspect intentionnel. Il en déduit les implications sur la conception du système technique. Cet examen et la projection des intentions appuient l'argumentaire en faveur d'une cible d'architecture.

c. Détermination par la technologie : la négociation logique-technique

En dépit de l'indépendance recherchée pour l'aspect logique à l'égard la technologie, certains choix de l'architecture technique peuvent conditionner les moyens d'expression de la modélisation logique. Le détail de ces moyens d'expression dépend des résultats de la négociation logique-technique. On ne se trouve pas tout à fait dans la même situation que pour les déterminations décrites précédemment. Alors que celles-ci influent sur le contenu et les décisions, la détermination par la technologie a un impact sur la forme, parfois dans les détails de l'expression. Le premier des procédés prévus par la méthode pour l'aspect logique s'intitule justement : « Arrêter les moyens de l'expression logique »[29]. L'exécution de ce procédé suppose un dialogue entre l'architecte logique et l'architecte technique. Cet exercice est critique pour la transformation du système technique. Il doit intervenir très tôt dans la trajectoire de transformation, avant de mobiliser lourdement les ressources de conception logique.

d. Détermination par l'aspect physique : l'anticipation du fonctionnement et de la performance du système technique

Le caractère abstrait de l'aspect logique n'exclut pas la préoccupation de la performance. En effet, si le modèle logique fait abstraction, autant que possible, des choix techniques, il tient compte, tout de même, d'éléments quantitatifs qui agissent sur le fonctionnement et la performance du système déployé. Il s'agit donc d'anticiper le comportement du système dans son aspect physique, où se joue l'exécution. Les informations à considérer incluent :

- la volumétrie et la fréquence des échanges ;
- la volumétrie des données et les temps d'accès et de mise à jour ;
- la fréquence des appels et la consommation de ressources ;
- les coûts d'exécution et de stockage.

[28] NB : valeurs et objectifs sont des éléments d'intention (faisant l'objet, respectivement, des modèles axiologique et téléologique).

[29] Référence du document : PxPCD-51.

Ces éléments ne sont complètement connus – et documentés – que dans l'architecture physique. Néanmoins, dès l'aspect logique, l'architecture peut s'appuyer sur :

- les modèles amont qui doivent eux-mêmes être quantifiés (le « métier » est le mieux à même d'indiquer le nombre d'instances d'une classe sémantique et la fréquence de création ; de même pour les métriques d'activité) ;
- des indications générales quant à la performance physique et à ses limites.

La description logique apporte ses propres informations quantitatives : le volume d'une structure d'échange, le graphe d'appel pour exécuter une fonction, etc. En combinant ces différentes sources – métier, logique, physique –, l'architecte peut anticiper le comportement du système, au moins dans les cas extrêmes. Il en tire les conclusions qui peuvent l'amener à retoucher l'architecture logique, voire la conception logique. Pour illustrer ces décisions, mentionnons l'impact sur la signature des services, la pagination des réponses, l'agrégation de résultats pour limiter le nombre d'appels ou, au contraire, le morcellement des résultats pour réduire le volume des flux…

3. Description de l'aspect logique

Le chapitre précédent traitait de l'aspect logique, en lui-même : sa nature, son contenu, les principes devant orienter son approche en vue de la qualité du système. Il est temps de nous doter d'un outil de représentation qui puisse nous aider à décrire l'aspect logique avec suffisamment de précision et de rigueur.

3.1 Exigences de représentation

L'illustration donnée de l'architecture fonctionnelle (figure PxPRD-50_5) montre les limites de ce type de représentation : elle organise des fonctions sur plusieurs niveaux, sans rien dire de ce qui se passe entre elles. En effet, les liens ne sont pas représentés. Or, découper un système fait nécessairement apparaître des relations, dépendances et interfaces.

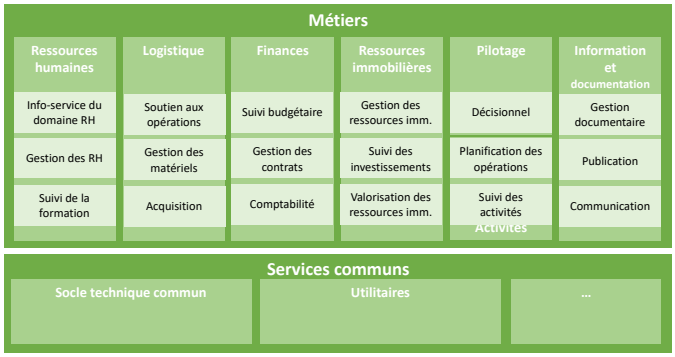
On ne peut parler d'exercice d'architecture qu'à partir du moment où l'on peut discuter des dépendances, des circulations, de la redondance, du couplage…

Bref, la représentation doit être suffisante pour soutenir un raisonnement architectural, c'est-à-dire pour évaluer la qualité structurelle. Pour cela, on ne peut pas se contenter de représenter les blocs et leur imbrication : le minimum est de représenter les dépendances et de faire apparaître les interfaces nécessaires. Il faut aussi suffisamment d'informations sur le système pour juger de la distribution des objets et des données.

L'architecte sera conduit à élaborer et jauger plusieurs scénarios d'architecture. Il doit décrire ces scénarios avec assez de précision pour pouvoir comparer leurs avantages et inconvénients. Il ne peut donc pas se contenter d'un vague schéma. Cette remarque n'enlève rien à l'intérêt de moyens comme les présentations ou les plans d'occupation des sols (POS) dans la tradition française de l'urbanisation du SI. Ces moyens de communication, certes utiles, ne sauraient remplacer un vrai dossier d'architecture logique. La documentation logique constitue le plan du système technique à construire ou à faire évoluer. Si nous nous entendons pour dire qu'il s'agit d'un système complexe, alors nous devons admettre aussi que sa description doit être à la hauteur de cette complexité.

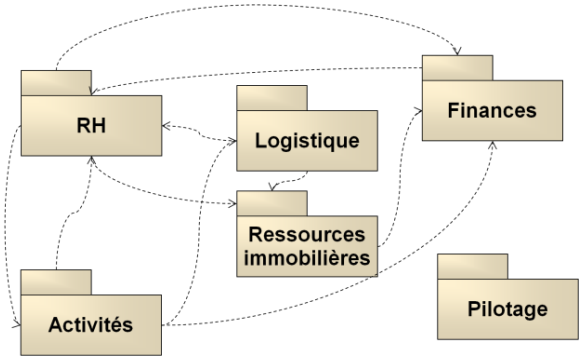
Figure PxPRD-50_16. Révélation de la complexité cachée dans un POS

Point de départ : POS



Extrait d'un plan d'occupation des sols classique : découpage en domaines fonctionnels (pour la plupart des blocs), pas de représentation des dépendances. Cet exemple s'inspire d'un cas réel.

Analyse des dépendances



Au cours d'un atelier avec les urbanistes de SI, au bout d'une demi-heure d'analyse, les dépendances apparaissent en soulevant la question de la distribution des objets dans l'architecture.

Le diagramme de droite révèle les dépendances entre les blocs logiques. Toutes les horreurs imaginables en architecture logique s'y rencontrent : relations mutuelles, bloc isolé (typique du « système décisionnel »), transitivité. Le couplage global du système fait froid dans le dos. Est-ce vraiment le système que l'on souhaite ?

Dans cet exercice, il s'agit de traquer les dépendances entre les blocs ou constituants. À partir de là, il sera possible de réarranger la substance et de proposer une meilleure architecture. Il convient d'examiner toutes les natures de dépendance :

- les flux ou flots d'information : informations qui passent d'un bloc à l'autre ;
- les appels ou flots de contrôle : blocs sollicitant l'exécution d'un traitement dans un autre bloc ;
- les références : désignations, par un bloc, d'éléments disposés dans d'autres blocs ;
- les ressources : partage des mêmes ressources utilisées par plusieurs blocs.

La première exigence en matière de représentation, celle sans laquelle on ne peut pas parler d'architecture, est donc, de toute évidence, la représentation des dépendances, en plus de celle des constituants.

Une autre exigence, plus générale, est la traçabilité, déjà évoquée. Il est très facile de la documenter : il suffit d'introduire un lien de traçabilité entre deux éléments. Encore faut-il que ces éléments – toujours de natures différentes – coexistent dans le même outil. Cette exigence renvoie à la notion de référentiel de description de l'entreprise (RDE). Le RDE se matérialise sous la forme d'une base de modélisation multi-aspect, intégrant plusieurs formalismes.

Puisque notre approche a une double portée, architecture (globale) et conception (locale), il importe que l'outillage distingue deux niveaux de spécification : externe et interne. En effet, dans la démarche de transformation, certains éléments devront être publiés très tôt et largement partagés : la spécification externe joue ce rôle, par opposition à la spécification interne qui concerne l'équipe ou le projet développant un constituant. Cette distinction n'a pas cours dans toutes les approches ou styles, mais on espère que l'outil de représentation la permettra.

Enfin, une exigence qui devrait aller de soi est celle de la continuité de la représentation. Nous entendons par là que l'outil de représentation qu'utilisent les architectes pour la vue synthétique devrait être le même que celui utilisé à tous les niveaux de la description logique, jusqu'au dernier détail. La continuité de représentation offre plusieurs avantages :

1. économique, d'abord, puisqu'elle réduit les frais de licences et d'équipement, et qu'elle évacue les tracasseries liées à l'interfaçage de différents outils ;
2. pratique, ensuite, en permettant un double mouvement du détail à la synthèse, et inversement, en s'arrêtant à tous les niveaux nécessaires pour une bonne appréhension ;

3. pédagogique, enfin, car tous les acteurs impliqués dans l'aspect logique communiquent dans les mêmes termes et se forment au même langage, lexique et syntaxe compris.

3.2 Choix de la notation

Nous cherchons donc une technique de représentation qui couvre toutes les catégories syntaxiques recensées dans le chapitre 2, qui satisfasse les exigences formulées ci-dessus, et qui, de plus, puisse s'adapter aux différents styles que nous aurons à traiter. Ce dernier point est, peut-être, un peu délicat : chaque style vient avec sa métaphore, sa terminologie et ses règles syntaxiques. Tout en assimilant ces variantes, la notation doit se montrer suffisamment formelle pour nous aider à :

- vérifier la conformité syntaxique ;
- produire des expressions « bien formées » conformément aux règles d'architecture ;
- porter les informations quantitatives dont nous aurons besoin pour juger d'une architecture proposée.

Inutile d'aller bien loin : le choix pragmatique que nous avons déjà entériné pour les aspects sémantique et pragmatique convient parfaitement. La notation UML, standard international, répond à toutes ces exigences, y compris celle de l'adaptation à un style particulier. En effet, la technique des profils UML permettra d'injecter dans l'outil, à moindre coût, la terminologie et les règles propres à un style particulier[30].

Certes, les évolutions d'UML l'ont affreusement compliqué, sans apporter le moindre éclaircissement méthodologique, bien au contraire[31]. On a le sentiment que chaque adjectif applicable à une notion donne lieu à une nouvelle méta-classe. On peut même douter de la cohérence de l'ensemble.

Longtemps, des notations dédiées à l'architecture ou à l'urbanisation du SI ont eu cours. Récemment, la notation ArchiMate est apparue, associée au référentiel de pratiques TOGAF. Demandons-nous si elles respectent les exigences formulées ci-dessus. On pourra objecter qu'elles apportent le vocabulaire qui manque à l'architecture, mais n'importe quel outil respectant le standard UML et celui des profils peut absorber facilement ce vocabulaire[32].

L'adoption d'un standard international présente, en outre, l'avantage d'une offre concurrentielle, écrasant les prix.

D'autres notations se présentent, au moins pour des domaines spécialisés : SysML pour l'architecture de systèmes, SoaML pour l'approche orientée services (présentée dans PxPRD-51s).

---

[30] Cf. PxPRD-51s pour l'orientation services. On y trouve une discussion relative à la notation SoaML.

[31] L'histoire d'UML peut se résumer comme suit : la première version, produite par les auteurs rassemblés par Rational, jetait les bases et manquait de précision. La deuxième version, avec un panel d'auteurs élargis, a consolidé le travail initial, mais en perdant de vue l'objectif de la pratique. Ensuite, la porte s'est grande ouverte à toutes sortes de courants et d'inspirations, avec des participants prompts à inscrire leur nom sur le standard, sans que personne ne se soucie de la cohérence d'ensemble, ni de l'applicabilité. Cette dérive a découragé beaucoup de praticiens, déjà rebutés par le manque de pédagogie et la tare originelle d'UML : le standard ne comporte pas de mode d'emploi, pas de méthode, pas d'explication quant à la mise en œuvre. Très tôt dans cette trajectoire, les éditeurs d'outils de modélisation se sont retrouvés face à la nécessité de « simplifier » le méta-modèle. À chaque nouvelle parution, ils commençaient par éliminer des éléments du standard avant de prendre en compte les évolutions dans leur outil. Aujourd'hui, nous en sommes au point où aucun outil ne respecte strictement, totalement, la spécification ; et l'on ne saurait le leur reprocher. D'ailleurs, qui utiliserait la totalité de la notation ? Le drame est qu'après une décennie d'effervescence (les années 90), consacrée par la standardisation ISO en 1997 de la version 1.5 d'UML, les pratiques de modélisation se sont lentement dégradées et ont presque disparu. Cette déliquescence s'accélère du fait du désintérêt du management pour ces questions. Il en résulte un gaspillage éhonté des ressources et des risques considérables pour l'avenir de nos systèmes.

[32] La manœuvre se révèle même trop facile, si l'on en croit la floraison spontanée de stéréotypes, en dehors de toute réflexion méthodologique.

3.3 Correspondance entre catégories de représentation et types d'éléments

UML propose une grande quantité de types d'éléments de modélisation, tous rigoureusement définis dans son méta-modèle. Cette section montre comment les catégories nécessaires à l'aspect se traduisent dans les termes d'UML. Dans ce document, nous nous plaçons toujours avant le choix d'un style.

a. Constituants logiques

La notion la plus fine, pour un constituant logique, pourra être reprise par la *classe* UML ; la plus large, par le *paquetage*, mécanisme général pour ranger les choses. Ainsi, la strate sera représentée par un paquetage.

Une notion plus riche et plus proche de la traduction informatique est celle de *composant*, en UML version 2. Notamment, un composant affiche des *interfaces*, fournies et requises, via des points d'accès. Le composant UML évoque plus naturellement le composant logiciel, appartenant à l'aspect logistique. L'architecture logique fera, de cette notion, un usage bien à elle.

Les traitements, quel que soit leur niveau, du bout de code réutilisé (façon « service ») jusqu'au programme lourd, peuvent se représenter par des *opérations* sur des classes ou des interfaces. Les composants UML peuvent aussi porter des opérations.

Les opérations en UML possèdent des *paramètres* en entrée et en sortie, ainsi qu'un résultat. Elles peuvent donc servir à décrire des fonctions, des opérations ou des services. Les paramètres fournissent un moyen économique d'exprimer des flux, quand on veut les associer à un traitement.

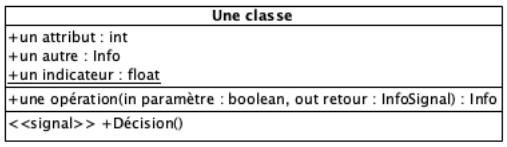


Figure PxPRD-50_17. Les types d'éléments pour la représentation des constituants logiques : la classe (indication du pouvoir d'expression)

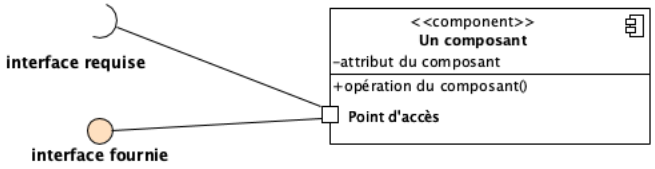
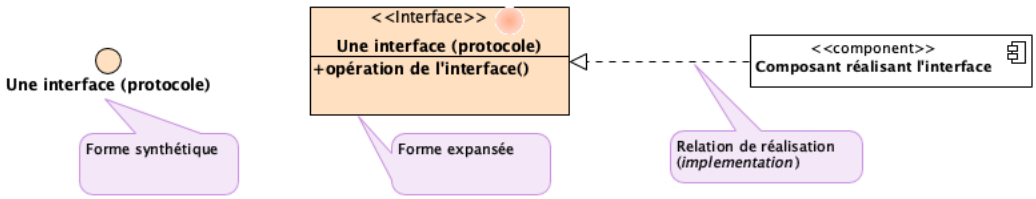


Figure PxPRD-50_18. Les types d'éléments pour la représentation des constituants logiques : le composant (indication du pouvoir d'expression)

b. Interfaces

Certains styles réifient les interfaces, c'est-à-dire traitent les relations entre constituants elles-mêmes comme des constituants d'un certain type. Si nous optons pour un tel style, comme c'est le cas avec SOA et l'API Management, nous utiliserons la notion d'interface établie par UML. La méthode pour l'orientation services fait grand cas de cette notion. Elle y trouve un moyen particulièrement efficace pour traiter le couplage.

Figure PxPRD-50_19. La représentation des interfaces réifiées

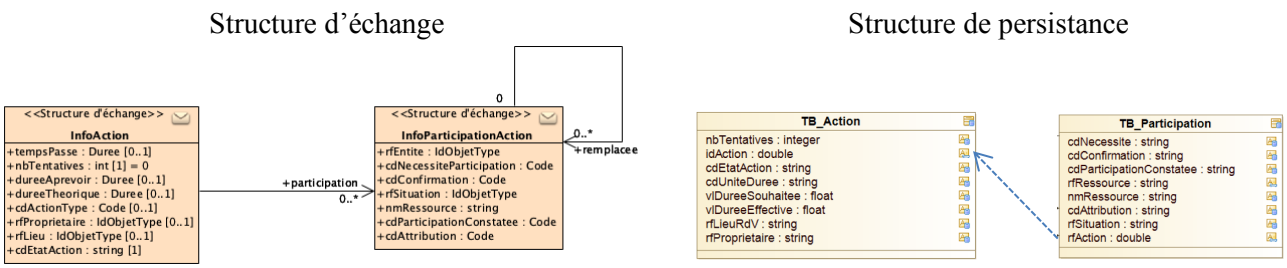


c. Structures d'information

L'aspect logique contient également des structures qui spécifient les supports de persistance et le format d'échange, respectivement : les structures de persistance et les structures d'échange. Pour les décrire en UML, on aura le choix entre les types de données (*data types*) ou les classes[33].

[33] Ce point est abordé dans les procédés PxPCD-53 et PxPCD-54.

Figure PxPRD-50_20. Exemples de structures d'information



d. Relations et communications

Bien entendu, la technique de représentation doit montrer tous les types de relations que nous sommes susceptibles de découvrir entre les constituants d'un système. Elles se rangent en quatre catégories :

- appel, soit le déclenchement d'un traitement à partir d'un autre ;
- flux, soit la circulation de l'information ;
- imbrication, soit la relation d'un composant à son composé (symétriquement : la composition) ;
- importation, soit l'incorporation d'un constituant dans un autre qui lui est extérieur.

Les deux premières relations se réalisent lors de l'exécution ; les deux dernières, au moment de la construction d'un constituant.

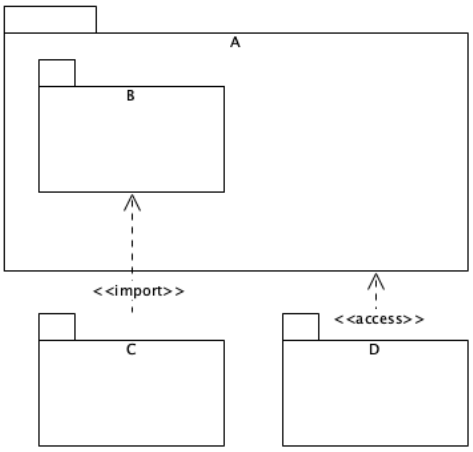


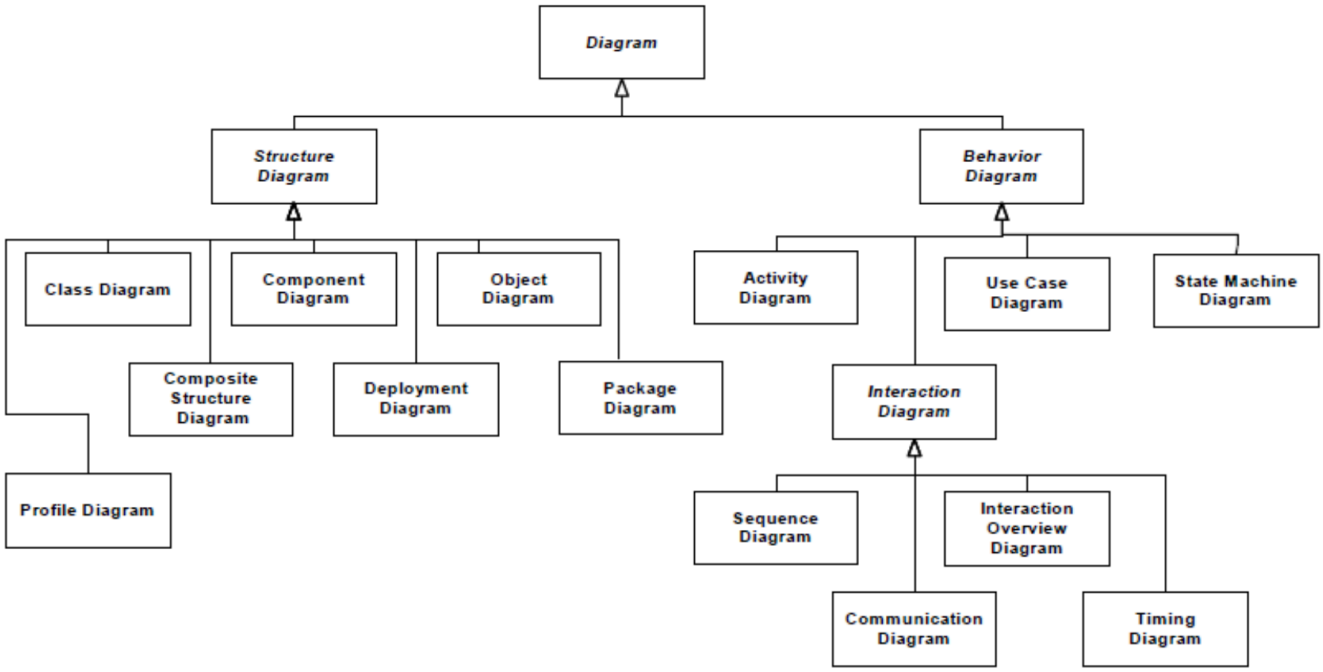
Figure PxPRD-50_21. Les relations de construction : quelques-uns des types de relations entre paquetages en UML

e. Diagrammes et graphes d'architecture logique

La spécification UML prescrit les quatorze types de diagrammes. Elle les ordonne selon l'opposition classique entre statique et dynamique, dans ses termes : structurel et comportemental (voir figure suivante[34]). À la place de cette division, nous préférons la tripartition indiquée au début du paragraphe 2.3 (p. 16). Elle présente l'avantage de donner plus de relief à cet outil particulièrement puissant qu'est la notion de machine à états.

[34] Cette figure est extraite de la spécification UML, version 2.5, p. 683.

Figure PxPRD-50_22. La taxinomie des diagrammes UML



Le tableau suivant indique, parmi ces types de diagrammes, lesquels contribuent à la description de l'aspect logique. Certains d'entre eux se rencontrent dans d'autres aspects, mais alors leur mode d'emploi diffère.

Figure PxPRD-50_23. Le recours aux types de diagrammes UML pour la description logique

Type de diagramme	Utile pour la description de l'aspect logique	Commentaire
Diagramme de profil	Méta-modèle	Utilisé pour injecter la méthode dans l'outil. La terminologie accompagnant le style retenu se traduit sous la forme de stéréotypes. Ce diagramme est dans les mains du méthodologue.
Diagrammes de classes	Pour les trois facettes	Diagramme utilisé dans les modèles logiques de données et d'échange. Il permet aussi de documenter le contenu des blocs, avec les constituants les plus fins.
Diagramme de structure composite	Non	Les procédés proposés pour l'aspect logique ne recourent pas à ce type de diagramme.
Diagramme de composants	Pour la facette des constituants	Ce type de diagramme permet de montrer les composants avec leurs interfaces, ainsi que les relations. La plupart des graphes d'architecture logique utilisent ce moyen.
Diagramme de déploiement	Non	Réservé à l'aspect physique (la notion de déploiement appartient à cet aspect).
Diagramme d'objets	Ponctuellement	Toujours utile pour illustrer le fonctionnement du modèle (par exemple : construction d'un flux complexe ; contexte d'exécution…).
Diagramme de paquetages	Oui (niveau architecture)	Pour les premiers niveaux de la décomposition du système ou d'une fédération de systèmes (graphes d'architecture de premier niveau).
Diagramme d'activité	Oui	Technique algorithmique pour spécifier le contenu des opérations.

Type de diagramme	Utile pour la description de l'aspect logique	Commentaire
Diagrammes d'interaction	Ponctuellement	Pour illustrer le fonctionnement dans des contextes donnés.
Diagramme de cas d'utilisation	Non	Uniquement dans l'aspect pragmatique, pour décrire des situations élémentaires de travail ou d'interaction avec l'entreprise.
Diagramme d'états	Oui	Les machines à états des aspects amont sont dérivées et reprises, si possible, par des automates logiques, inclus aux constituants.

3.4 Modèles logiques

Cette section introduit les types de livrables attendus de la conception logique. Ces livrables doivent couvrir les trois facettes de l'aspect logique : échange, substance, persistance. Chaque facette possède sa propre logique, ses propres contraintes. Il est donc nécessaire de les décrire de façon distincte[35].

Pour que la suite se comprenne bien, nous devons revenir sur la situation du modélisateur :

1. L'idée de base repose sur l'existence d'un unique référentiel de description de l'entreprise (RDE), dans lequel se déverse tout élément de description concernant le Système Entreprise, y compris les spécifications logiques.
2. Cette solution s'impose (du moins, elle est à privilégier) pour encourager une conception détaillée en cohérence avec les objectifs globaux de l'architecture. Toute la difficulté de la démarche consiste à maintenir la conception dans le cadre posé par l'architecture. Cette discipline devient plus naturelle et mieux contrôlée quand les concepteurs interviennent dans la même base de modélisation, le référentiel. En effet, ce dernier se structure d'après les décisions d'architecture. Il reste au concepteur à « remplir les cases », si l'on peut dire.
3. Par ailleurs, en pratique, il n'est pas question de donner à lire, au niveau des projets, le contenu complet de l'aspect logique. La documentation est générée partiellement, pour les besoins d'un projet. Ce sont ces extraits, ciblés sur un objectif et confinés à un périmètre, que nous appelons les modèles. Un modèle est donc un extrait de la description complète du système.

Comment prescrire le contenu attendu d'un modèle ?

On pourrait opter pour un modèle qui couvrirait toutes les facettes, dans un périmètre donné. Mais il est tout de même préférable de dissocier ces facettes dans la documentation, parce qu'elles évoluent de façon séparée. Par ailleurs, la diffusion diffère également : le modèle des échanges concerne tous les utilisateurs potentiels de ce qu'expose un constituant, tandis qu'un modèle de données, dans les nouvelles approches, a de bonnes chances d'être plongé au cœur d'un constituant et ne sera lu que par les concepteurs et développeurs dudit constituant.

Les modèles sont établis pour un périmètre défini, pas forcément l'ensemble du système. En général, ils couvrent les responsabilités d'un constituant délimité dans l'architecture logique : un sous-système, un agrégat. Cette couverture peut être partielle, circonscrite aux besoins et possibilités d'un projet. Cependant, la démarche sera d'autant plus efficace que le modèle approche l'exhaustivité, en anticipant les besoins futurs.

Il arrive aussi que le modèle soit défini par rapport à un périmètre fonctionnel, associé à un objectif de projet classique. Dans ce cas, le modèle présentera un contenu distribué sur les constituants nécessaires pour réaliser cette fonction. L'affaire devient plus délicate. Les choses ne pourront bien se passer qu'à la condition d'une architecture fortement affirmée (voir le § 4.4).

[35] On n'a rien dit de plus idiot, en génie logiciel, que l'approche « *code first* » : un seul modèle suffirait, le code ! Et pourtant, Dieu sait si l'on en dit, des idioties, dans ce domaine.

a. Modèles logiques des échanges

Le modèle logique des échanges assemble les structures d'échange. La documentation complète de celles-ci constitue la spécification du langage pivot. Pour atteindre cet objectif et produire un vrai langage pivot dont on attend interopérabilité et universalité, la conception doit respecter des règles strictes. Ce travail fait l'objet du procédé PxPCD-53.

Pour documenter les structures d'échange, on peut hésiter entre les types de données (*data types*) et les classes. Les premiers suffisent, mais le choix dépend de l'outillage et de ses capacités de génération. Par exemple, les structures d'échange peuvent se convertir en XSD, si telle est la cible au niveau logiciel. Les outils de modélisation sont capables de les générer automatiquement, certains à partir de classes, d'autres à partir de types de données.

Les structures d'échange se relient par des associations à la sémantique appauvrie, limitée à un petit nombre de relations syntaxiques telles que :

- l'imbrication (des flux dans des flux) ;
- la référence ;
- bien sûr, la traçabilité.

Pour les représenter, le diagramme de classes fera l'affaire.

b. Modèles logiques des constituants

La documentation des constituants se divise en deux parties :

1. la spécification externe, couvrant tout ce qu'il faut connaître du constituant, vu de l'extérieur, pour avoir l'idée de l'utiliser et en connaître le mode d'emploi ;
2. la spécification interne, comportant tous les détails, jusqu'aux algorithmes et machines à états, pour la réalisation du constituant, hors les choix techniques.

La démarche projet doit mettre clairement en place cette division. Dans la base de modélisation, elle se matérialise à travers les types d'éléments. Par exemple, la spécification externe porte sur les premiers niveaux d'agrégats et sur les interfaces, tandis que la spécification interne rentre dans le contenu des derniers niveaux d'agrégats. La division s'exprime alors par des procédures de génération documentaire qui filtrent les éléments selon leur type.

c. Modèles logiques de persistance

Il s'agit des classiques modèles logiques de données (MLD). Pour les élaborer, il faut déjà avoir choisi un style de persistance. Cette décision a été occultée, longtemps, par la prédominance des SGBD relationnels. Aujourd'hui, l'offre technologique se diversifiant (notamment avec le NoSQL), la question revient. Du style de persistance découlent :

- les règles de dérivation à appliquer aux modèles amont ;
- les formes que prend le MLD.

Le procédé PxPCD-54 porte sur l'élaboration du modèle logique de persistance.

3.5 Documentation de l'architecture logique

Le point de vue global sur le système étudié s'exprime dans deux catégories de livrables :

1. les premiers fournissent les plans du système à construire ou à faire évoluer ;
2. les seconds apportent, aux projets, l'éclairage de l'architecture et les consignes à respecter pour que leurs efforts contribuent à l'édification de la maison commune.

a. Dossier d'architecture logique

Documenter l'architecture de l'aspect logique est un préalable à tout effort de création ou de transformation du système technique. Cette documentation prend la forme d'un « dossier d'architecture logique » qui rassemble les décisions structurantes, à commencer par le choix du style et sa justification.

Le terme « dossier » indique que ce livrable s'inscrit dans la durée. Contrairement aux modèles évoqués ci-dessus, le dossier d'architecture logique accompagne le système pendant toute la trajectoire de transformation et au-delà. Il connaît donc plusieurs versions :

1. La première version jette les fondations, fixe la structure d'ensemble. On espère qu'elle couvre les décisions d'architecture logique à 80%. Malgré le soin que l'architecte y apporte, cette première version ne peut anticiper toutes les évolutions, ni traiter la totalité du périmètre au niveau de précision requis pour une architecture maîtrisée. Dans les styles qui s'y prêtent, la priorité est l'identification des interfaces. Un dossier d'architecture logique qui identifie, en première version, 80% des interfaces avec leur mandat, constitue une cible raisonnable. Il mettra les investissements sur de bons rails, pour les années à venir.
2. Annuellement, les décisions budgétaires et de planification motivent une révision du dossier. Les thèmes laissés flous font alors l'objet de précision, si le planning des investissements prévoit de les aborder bientôt.
3. Enfin, l'accompagnement des projets par l'architecte logique peut soulever de nouvelles questions et déboucher sur une consolidation du dossier. Par exemple, une interface doit être ajoutée, voire un constituant.

Le corps de ce livrable décrit et justifie la structure du système à construire. Il établit le plan d'ensemble auquel les projets devront se conformer. Il s'augmente des décisions qui pourront être prises au fil du temps.

Le dossier d'architecture logique se place sous la responsabilité de l'architecte logique, qui assume son contenu et sa mise en œuvre. Il doit recevoir également la validation et l'appui effectif des décideurs, non seulement les responsables techniques (pour un système informatique, le directeur informatique), mais aussi la direction générale. La raison est la suivante : les travaux à l'échelle du système technique revêtent de tels enjeux et appellent un tel niveau d'investissement, dans la continuité, que sans l'aval et l'engagement de la plus haute autorité, ils n'ont aucune chance d'aboutir. Cet engagement viendrait-il à manquer qu'il ne s'agirait plus d'architecture, mais, au mieux, d'entretien et de sauvegarde des apparences.

b. Consignes aux projets

Dans une démarche guidée par l'architecture, la difficulté n'est pas tant d'élaborer les plans détaillés du futur système, que de canaliser l'énergie mise dans les projets. À rebours de la culture actuelle, dominée par le mode projet, il est urgent de restaurer l'autorité de la fonction « architecture », afin que les projets travaillent à l'œuvre commune. La rationalité économique autant que les visées stratégiques l'exigent.

Matériellement, ce volontarisme se traduit par des consignes précises que l'architecte logique doit être capable de formuler, puis de faire respecter. Cette capacité repose sur deux piliers : la compétence et l'autorité. Le présent guide ambitionne d'asseoir la compétence sur une perception de la dimension « Produit ». Quant à l'autorité, c'est une question de gouvernance, à traiter dans la dimension « Processus » de la méthodologie[36].

Tout particulièrement, au début du projet, l'architecte logique analyse le périmètre du projet et en déduit les constituants de la solution. Ceux-ci préexistent, ou sont prévus dans le plan, ou encore ils doivent être complétés.

[36] Cette opposition Produit/Processus renvoie au schéma Pro^3, structurant la méthodologie. Cf. le Guide général, PxMDS-01.

En tout cas, le projet n'est pas libre de la forme que prendra la solution qu'il doit livrer. Pour l'essentiel, la solution s'obtient par l'assemblage de constituants identifiés dans la cible d'architecture.

L'architecte produit donc un graphe d'architecture logique qui indique les constituants intervenant dans le périmètre du projet, certains à réemployer, d'autres à développer. Des explications, commentaires, recommandations… complètent ce graphe. Le tout forme le dossier d'architecture logique appliqué au projet.

Le projet doit suivre ces consignes, exactement comme une équipe de BTP suit les plans et ordres de l'architecte.

Une organisation qui ne serait pas capable d'appliquer cette règle de bon sens se condamne au gaspillage.

Si l'architecte logique a pu passer suffisamment de temps sur sa planche à dessin et a produit un vrai dossier d'architecture logique (tel que décrit au paragraphe précédent), alors il ne lui faut pas longtemps pour en extraire la matière et pour formuler les consignes dans le cadre d'un projet particulier.

Cette nouvelle dynamique est facilitée quand les pratiques des uns et des autres trouvent à s'appuyer sur une méthodologie commune.

3.6 Qualité de la conception logique

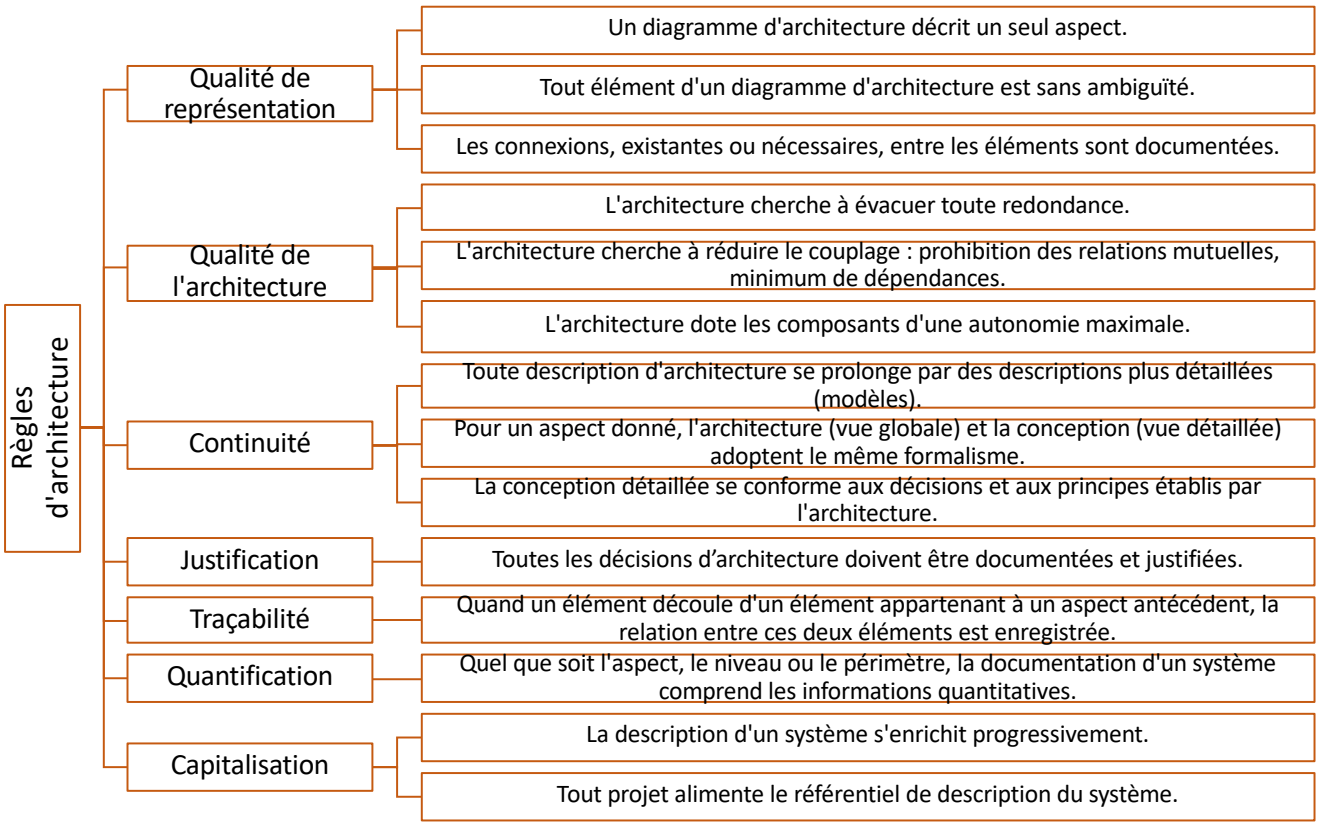
Ce chapitre sur la description de l'aspect logique se conclut sur les critères d'acceptation applicables aux livrables. Nous les exprimons sous la forme de règles d'architecture, organisées en sept thèmes.

À partir de ces règles, il est possible de définir des indicateurs. Nous pouvons donc envisager une métrologie de l'aspect logique. Elle apporte sa contribution à la métrologie de l'entreprise[37].

Il est à noter que beaucoup des règles données ci-dessous s'appliquent à d'autres aspects ou peuvent se généraliser.

[37] Sur la discipline de la métrologie d'entreprise, voir la série de procédés PxPCD-13.

Figure PxPRD-50_24. Récapitulatif des règles d'architecture



a. Qualité de la représentation

Un diagramme d'architecture ne décrit qu'un seul aspect.

Méfions-nous des schémas qui mélangent « fonctionnel » et « applicatif », ou « métier » et « informatique », etc. Certes, il peut être utile, parfois, de montrer sur une même diapositive des préoccupations de niveaux différents. On parle, alors, de « vues », établies dans une intention de communication précise. Elles n'ont rien à voir avec une représentation d'architecture.

Rappelons que le premier de tous les principes est celui de la séparation des niveaux de préoccupation. Ce principe ne fait que reformuler un précepte de bon sens : face à la complexité, commençons par sérier les décisions.

Un schéma, donné comme schéma d'architecture, et qui mêlerait plusieurs aspects, entraînerait la confusion. À coup sûr, il décrirait insuffisamment chacun de ces aspects.

Chaque aspect se caractérise par une matière propre ; il possède sa propre logique, ses règles particulières. Les représentations professionnelles doivent donc séparer les aspects, ne serait-ce que pour leur appliquer les moyens d'expression qui leur conviennent.

Tout élément d'un diagramme d'architecture doit être sans ambiguïté.

Chaque aspect contient des éléments qui se rangent sous un nombre limité de types. Pour l'aspect logique, le chapitre 2 a commencé le recensement. Les guides et procédés dédiés à des styles singuliers le complèteront. Une fois le méta-modèle établi, le modèle de l'aspect logique ne saurait contenir des éléments d'un autre type.

De la même façon, les fantaisies que permettent les outils de communication, comme de placer un élément à cheval sur plusieurs autres, ne peuvent subsister dans la documentation formelle.

L'ambiguïté pourrait naître des dénominations utilisées. Il va de soi que le précepte général de la modélisation s'applique à l'aspect logique :

Une même désignation ne peut s'appliquer qu'à un seul élément, dans la totalité du système, perçu sous son aspect logique.

L'exigence de documentation ne s'arrête pas aux nœuds de l'architecture, les choses évidentes que sont les constituants ; elle porte également sur les liaisons, relations, connexions. D'où une autre règle applicable à la description :

Les connexions, existantes ou nécessaires, entre les éléments sont documentées.

Non seulement les dépendances apparaissent sur les graphes d'architecture, mais aussi elles font l'objet de commentaires : discussions, justifications, quantifications. Sans ce niveau de description, comment pourrions-nous soutenir un raisonnement d'architecture et évaluer la qualité structurelle ?

b. Qualité de l'architecture

Une fois assurée la qualité de la représentation, le raisonnement architectural peut commencer. En effet, l'architecte logique a besoin de s'appuyer sur la représentation, plus encore que l'architecte en bâtiments : celui-ci établit les plans pour décrire quelque chose qui n'existe pas et qui sera construit ; celui-là, même intervenant sur un système technique existant, ne peut l'appréhender spontanément, car une composante importante de ce système est abstraite, immatérielle. Cette caractéristique est évidente dans le cas des systèmes informatiques, mais même dans les autres cas, la part de plus en plus grande que prend le logiciel – le *firmware*, l'informatique embarquée – rend l'appréhension difficile. De toute manière, la complexité de nos systèmes excède les capacités mentales d'un être humain. Il faudrait être fou et irresponsable pour s'imaginer pouvoir maîtriser ces systèmes sans une représentation rigoureuse.

Le plan du système, suffisamment formel, révèle ses propriétés et sa structure. Sans surprise, la première règle précise un des principes directeurs donnés dans le chapitre 2 :

L'architecture logique cherche à évacuer toute redondance.

Cet objectif vaut pour l'aspect logique, la redondance ayant des manifestations et des impacts différents selon l'aspect. Une architecture logique idéale montrerait un taux de redondance nul. Cette qualité n'est pas si simple à mesurer, car la redondance peut se cacher derrière le vocabulaire. Une métrique assez évidente est le nombre de traductions logiques données à un élément d'un aspect « métier ». La valeur cible devrait être inférieure ou égale à 1.

L'autre objectif de l'architecture logique s'apprécie plus facilement, à partir d'une représentation formelle :

L'architecture cherche à réduire le couplage au sein du système, à son strict nécessaire.

La structure optimale d'un système est celle qui ne conserve que le couplage nécessaire au bon fonctionnement. Cette visée peut entrer en contradiction avec l'élimination de la redondance. En effet, celle-ci conduit à confier les informations d'une certaine nature à un seul constituant ; les constituants ayant besoin de ces informations doivent alors se connecter à ce constituant, unique propriétaire des informations. L'architecture logique équilibre ces deux objectifs. Elle donne la priorité à la non-redondance, tout en sachant que d'autres choix pourront être faits dans les aspects logistique et, surtout, physique.

L'analyse du couplage comprend :

- une perspective qualitative : nombre et localisation des dépendances, lesquelles établissent la structure ;
- une perspective quantitative : fréquence et volume des interactions qui se jouent aux interfaces entre les constituants.

D'un point de vue mathématique, l'architecture d'un système est un « graphe valué » (les arcs sont pondérés).

c. Continuité

Un graphe d'architecture se comprend comme une synthèse de la description complète du système, fixant la structure de celui-ci. Réciproquement, les représentations des derniers niveaux de détail s'inscrivent obligatoirement dans la structure d'ensemble.

Cette continuité traduit une exigence de cohérence. Elle s'imposerait avec la force de l'évidence si tous les acteurs impliqués respectaient les règles du jeu entre le point de vue global de l'architecture et le point de vue détaillé de la conception. Malheureusement, la pratique est un peu différente. C'est pourquoi il est nécessaire d'insister sur cette règle. S'écarter de cette règle revient à accepter la gabegie d'investissements parcellaires, menés dans une approche court-termiste et augmentant la dette technique. Elle a pour conséquences :

- l'adoption d'un formalisme unique et, si possible, d'un même outil pour couvrir toute la chaîne d'activité, de l'architecture à la conception interne détaillée[38] ;
- le renforcement de l'autorité des architectes dans leurs relations avec les projets.

d. Justification

Les décisions d'architecture sont documentées et justifiées.

Il ne suffit pas de dessiner la structure du système ; il faut aussi l'argumenter et montrer en quoi la conception répond aux préoccupations de l'entreprise. Tout particulièrement, quand l'architecte logique décide de supprimer des dépendances reçues de l'architecture métier, il doit expliquer en quoi cette décision est possible. La réduction du couplage motive, bien sûr, ce type de décision. Encore faut-il qu'elle n'empêche pas le fonctionnement normal du système. La documentation de l'architecture doit conserver le raisonnement qui a abouti à la décision d'architecture.

[38] Cette conception détaillée se pare d'un nouveau titre, « architecture de solution », importé tout droit des USA, sans aucun questionnement. Cet emprunt est malheureux, puisqu'il affaiblit le terme « architecture ». La solution étant l'application, il convient de la structurer : c'est ce que l'on a toujours nommé « conception », en réservant le terme « architecture » pour la réflexion sur le système dans son ensemble. Par ailleurs, ce changement de titre – le concepteur devenu « architecte de solution » – va de pair avec une dérive vers les questions techniques. Résultat : trop peu d'attention est portée à la structuration.

e. Traçabilité

Quand une décision répond à une préoccupation générale (une intention) ou quand un élément logique dérive d'un contenu en amont, la documentation conserve la trace de cette relation.

Les retombées de cette règle se révèlent dans deux circonstances :

1. Son application contribue à justifier l'expression logique (règle précédente).
2. En cas d'évolution des éléments des aspects antécédents, l'analyse d'impact se mène très rapidement et facilite la maîtrise des transformations subséquentes.

f. Quantification

Quels que soient l'aspect concerné, le niveau ou le périmètre, la documentation d'un système comporte des informations quantitatives permettant d'éclairer les décisions.

Quelques exemples :

- Sur l'aspect sémantique : le nombre d'instances d'un concept, cumulé, par unité de temps, etc. ; la fréquence de changement de l'état.
- Sur l'aspect pragmatique : le niveau des consommations des activités, y compris le temps passé ; la fréquence d'apparition des événements.
- Sur l'aspect physique : les conséquences sur le stockage des données, les performances des réseaux et machines.

g. Capitalisation

Sous ce terme, il s'agit de :

- mettre en place une dynamique de consolidation de l'architecture logique ;
- décrire progressivement, sans rien perdre et avec le maximum d'efficacité, le Système Entreprise dans sa totalité.

Cette dynamique s'oppose à une organisation fondée essentiellement sur le mode projet, dans laquelle les efforts de description se bornent aux besoins immédiats des projets. La démarche doit s'inverser : d'abord, les plans du système dans son ensemble ; ensuite, le détail apporté au fur et à mesure des investissements et des interventions sur ce système.

Une conséquence pratique est que l'on évitera de doter les projets de leur propre base de modélisation. Le mieux est qu'ils alimentent directement la base de connaissance commune, de façon à l'enrichir en respectant la structure prescrite. Dans cette démarche, nous devons prévoir des ajustements à apporter à l'architecture logique, afin de répondre aux découvertes toujours possibles quand on attaque les travaux sur le terrain des projets.

L'architecture logique, établie préalablement à la transformation du système, se consolide néanmoins dans sa relation avec les projets.

La démarche appliquée au niveau des projets doit s'engrener dans la dynamique globale. La fonction première des projets est d'apporter leur pierre à l'édifice commun. D'où cette règle :

Tout projet significatif fournit des modèles a) conformes à l'architecture cible et b) qui enrichissent le référentiel de description de l'entreprise.

4. Utilisation des modèles logiques

Les livrables issus de l'aspect logique interviennent dans les circonstances suivantes :

1. la spécification des composants à réaliser dans le système technique ;
2. la négociation logique-technique qui résout l'équation entre l'indépendance de l'aspect logique et sa convertibilité en termes techniques ;
3. l'administration du système technique, sur le long terme, dans le souci de la valorisation des moyens investis ;
4. la transformation du système.

4.1 Spécification des composants techniques

L'approche de l'aspect logique a pour première retombée l'identification et la spécification des composants du système technique, notamment informatique. En faisant abstraction des détails techniques, elle met l'accent sur la structuration du système. Les questions soulevées par cette approche méthodique permettent d'échapper au conditionnement culturel et aux ornières qui marquent l'élaboration des grands systèmes techniques. Son résultat le plus évident consiste à découper les sous-systèmes – les constituants – d'une façon fort différente de ce que l'on trouve dans les systèmes existants. Ce résultat s'explique par la prise en compte de la globalité du système, de ses interactions avec son environnement, ainsi que par le souci de la qualité structurelle.

Peut-être la preuve la plus manifeste se trouve-t-elle dans la spécification du format d'échange. Pour arriver à un vrai langage pivot, il est indispensable de s'arracher à la logique projet et de considérer ce langage dans la totalité des échanges : au sein du système comme entre les systèmes.

Quand l'effort mis dans la conception logique va jusqu'à livrer une spécification détaillée des opérations, il ne reste plus qu'à la traduire dans l'aspect logistique, en intégrant les choix techniques.

Dans cette démarche, la réalisation devient un acte de traduction : il s'agit de traduire la spécification logique dans les termes de l'architecture technique cible.

Cette façon de faire, inspirée par MDA, sécurise les développements et améliore la gestion des compétences et des responsabilités.

4.2 Négociation logique-technique

L'indépendance de l'aspect logique par rapport à la technologie a été posée dès le début. Elle fonde la notion même d'aspect logique. Malgré tout, elle reste relative. Disons qu'il y a quelques précautions à prendre. Une vraie conception logique réclame des moyens et du temps. Il ne faudrait pas se retrouver avec des livrables de nature logique qui s'éloigneraient trop de la cible technique, au point où ils deviendraient inutilisables pour la suite des travaux.

La négociation logique-technique a pour but d'assurer les conditions de convertibilité de la modélisation logique.

Du point de vue logique, elle se conclut par le choix des moyens d'expression que les modélisateurs logiques pourront utiliser.

Ce travail cherche à équilibrer :

1. le souci de maintenir l'indépendance logique, garantissant la durée de vie de la description logique ;
2. la convertibilité technique, sans laquelle l'investissement sur l'aspect logique serait vain.

Ainsi, à chaque fois que surgit une nouvelle tendance technologique, se pose la question d'une rupture de paradigme. Le cas échéant, le changement de style entraîne la modification de l'expression logique, donc une double dépense : d'une part, il faut réviser les modèles ; d'autre part, il faut aussi réformer les compétences et les pratiques. C'est donc avec prudence que l'on décrètera un changement de style[39].

4.3 Administration des systèmes techniques

Étant donné le niveau des investissements nécessaires à la construction de nos systèmes techniques, nous leur souhaitons la durée de vie la plus longue possible. Pourtant, nous sommes pris dans un dilemme :

1. D'une part, faire fructifier l'investissement suppose d'inscrire l'évolution des systèmes dans la longue durée.
2. D'autre part, au fil des corrections et évolutions, la complication et l'entropie s'accroissent, menant à une perte de maîtrise et à une prise de risque de plus en plus préoccupante.

Face à cette situation, le calcul stratégique se fait en termes de dette technique et de risques encourus. La représentation logique lui offre un appui, une compréhension plus accessible. La mesure de la qualité structurelle tient une place essentielle dans cette analyse.

En outre, l'approche logique encourage l'effort de documentation, presque toujours insuffisant dans les pratiques générales, quel que soit le secteur d'activité et quel que soit le type de système[40]. D'ailleurs, cet aspect logique n'existe qu'à travers la documentation. On peut voir là un corollaire de sa situation intermédiaire. Spécialiser des compétences et distribuer des responsabilités sur cet aspect garantissent donc l'existence de la documentation. Celle-ci cristallise une part de l'investissement et la rend disponible pour de nouveaux travaux dans l'aspect logistique.

La description logique capture une part de l'investissement sur le système technique et en démultiplie les retombées, sur le long terme, par-delà les changements technologiques.

Inversement, quand l'investissement se réalise directement et uniquement sous la forme technique – typiquement le développement logiciel –, sa période de fructification se trouve limitée à l'espérance de vie de la technologie retenue.

---

[39] À titre d'illustration, la question s'est posée quand REST l'a emporté sur les standards Web Services (la pile des normes WS*). Fallait-il changer la façon de modéliser ? De même, les micro-services représentent-ils une approche concurrente par rapport à SOA ?

[40] On décrie souvent les systèmes de gestion pour la faiblesse de leur documentation. On serait horrifié de découvrir que ce faible niveau se retrouve dans des systèmes réputés complexes et sensibles, tels que les systèmes de transport ou d'armement. La forme même de la documentation est révélatrice : très souvent, la seule documentation disponible se présente sous forme de milliers d'exigences, stockées dans des classeurs, et, bien sûr, sans qu'il soit possible d'en montrer la structure et la redondance. Cet amateurisme pratique (ce « bricolage » disent certains sociologues) s'observe même dans des industries prestigieuses. Il a causé la perte de pans entiers de notre industrie.

4.4 Transformation des systèmes informatiques

Au-delà de l'administration du système, il nous faut envisager sa transformation. Les raisons de celle-ci peuvent être endogènes, liées à l'état du système technique, ou exogènes, à trouver dans les aspects amont ou la technologie elle-même.

Dans la perspective de la transformation, qu'apporte le point de vue logique par rapport à une approche purement technique ? Pourquoi, par exemple, ne pas se contenter d'une architecture applicative[41] ?

Tout d'abord, pour réellement transformer, nous devons prendre nos distances par rapport à l'état actuel du système.

Ensuite, la transformation réclame l'idée. L'architecture logique, à la fois par sa reprise de l'architecture métier et par sa portée globale, découvre les plans du système technique optimal. Ces plans, affranchis du poids de l'existant, proposent un « idéal régulateur » aiguillonnant la réflexion architecturale. Même quand cet idéal n'est pas conservé tel quel comme cible, il sert au moins d'étalon pour évaluer la réponse technique. Notamment, dans les cas de fédérations de systèmes (filiales, partenaires…), il montre avec l'éclat de l'évidence jusqu'où pourrait conduire une transformation ambitieuse, et quels services et économies pourraient en découler. Il est intéressant et relativement facile de montrer l'écart entre, d'un côté, l'architecture optimale et, de l'autre, l'architecture en place. De même pour une architecture cible, pas toujours très audacieuse. Révéler cet écart éclaire le raisonnement d'architecture et les décisions à prendre dans la trajectoire de transformation.

Toutes les bonnes idées popularisées aujourd'hui sous des slogans comme l'API Management ou l'orientation services n'ont aucune chance de se réaliser si l'impasse est faite sur l'aspect logique. Or, les enjeux pour les entreprises sont considérables, non seulement en termes financiers, mais aussi en termes d'opportunités.

Enfin, dans la tension entre les deux portées (globale/locale), l'approche logique apporte une aide considérable pour instituer la nouvelle dynamique de construction des systèmes, entre architecture et projets.

5. Conclusion et mise en œuvre

5.1 Compétences

À l'ère du numérique, tant vantée, les enjeux évoqués dans le premier chapitre revêtent une importance toute particulière. Isoler l'aspect logique conditionne le succès des transformations nécessaires pour adapter les entreprises à cette nouvelle donne.

Il est donc urgent de restaurer deux disciplines, battues en brèche par l'évolution des pratiques :

1. l'architecture logique,
2. la conception logique.

La réhabilitation de ces disciplines favorisera l'économie des compétences. En effet, dans ces domaines, les compétences et les pratiques s'appuient sur un corps de principes et de savoirs très stable, comme veut le montrer ce guide. Dans le détail, seul un changement de style oblige à un nouvel effort d'apprentissage et d'outillage. Or, ceci ne se produit qu'une fois toutes les deux ou trois décennies. En conclusion, les compétences sur l'aspect logique sont beaucoup plus stables et profitables que des compétences liées à la technique[42].

---

[41] L'expression « architecture applicative » peut être prise dans deux sens : a) l'ensemble des applications ; b) la structure du système logiciel, autrement dit l'architecture de l'aspect logistique. Dans les deux cas, elle porte sur le logiciel, par opposition à l'aspect logique. Une troisième acception possible est le choix d'un style d'architecture, décomposant le système en applications. Mais là, c'est une cause perdue !

[42] Les responsables de ressources humaines y verront un moyen positif de gérer des profils « senior », découragés par la versatilité des solutions techniques et frustrés de ne pas pouvoir mettre à profit leur expérience. Autrement dit, l'obsolescence guette les compétences et les individus, tant que l'on envisage les choses sur le plan strictement technique. Les disciplines de l'aspect logique prémunissent de cette obsolescence, pourvu que l'on reconnaisse leur rôle dans la maîtrise des systèmes.

5.2 Procédés liés à l'aspect logique

Ce guide s'est volontairement limité aux principes, règles et raisonnements valables sur l'aspect logique, quel que soit le style d'architecture retenu. Il ne suffit donc pas pour passer à la mise en œuvre de l'approche logique. Pour ce faire, il est nécessaire de préciser la terminologie et les critères dans le cadre d'un style précis. On trouvera ces compléments dans des guides spécialisés, à commencer par le guide PxPRD-51s qui traite des architectures orientées services.

Les guides méthodologiques forment le socle de connaissances sur lequel reposent les pratiques. Leur vocation est quelque peu ésotérique : ils se destinent aux méthodologues et architectes qui s'intéressent à la refondation des bonnes pratiques. Le détail des pratiques, concernant un plus vaste public, se trouve dans les fiches de procédés, la partie exotérique de la méthode. La fiche PxPCD-50 introduit la série de procédés portant sur l'aspect logique.

Figure PxPRD-50_25. Conseils de lecture

Code utilisé dans cette fiche	Titre, édition	Auteur	Commentaire
ASA	*Applied Software Architecture*	Hofmeiter, Nord, Soni	
BCF	*Business Component Factory*	Peter Herzum, Oliver Sims	Référence de l'approche par les composants
CAS	Le SI démystifié	Yves Caseau	
CES	*CESAM : CESAMES System Architecting Method*	CESAMES	http://cesam.community/articles-docs/publications-cesam-community/
MDA	*Model Driven Architecture*	OMG	Le standard
MEA	MDA en action	Xavier Blanc	Pour appliquer
MER	Merise	Tardieu, Rochfeld, Coletti	
GGEN	Praxeme « Guide général »	D. Vauquier	Disponible sur www.praxeme.org
SSA	*Software Systems Architecture*	Nock Rozanski, Edin Woods	
TACT	Architecture technologique des systèmes d'information – La méthode TACT	Nidal Alakl, Jean-Christophe Lalanne	Méthode d'architecture logique
UML	*Unified Modeling Language, version 2.5*	OMG	La notation standard
YRD	*Modern Structured Analysis*	Edward Yourdon	Parmi les ouvrages sur l'approche de conception structurée

Table des figures

Index

Table analytique