

↗ EA

CURSUS COMPLET

Architecture d'Entreprise.

De novice à sachant.

12 leçons · ~30 heures
Édition numérique · August 2026

Penser juste. Décider juste.

Sommaire .

L01	Dépasser le mythe technique
L02	La Phase Préliminaire : cadrer l'espace de jeu
L03	Le Cycle ADM : le moteur méthodologique
L04	La Vision (Phase A) : éviter les choix hâtifs
L05	L'Architecture Métier (Phase B) : la Business Capability Map
L06	Les Processus (BPMN) : l'action opérationnelle
L07	L'Architecture des Données : la pyramide DAMA
L08	L'Architecture Applicative : le découplage des systèmes
L09	L'Architecture Technique : les services de plate-forme
L10	Opportunités et Solutions (Phase E & F) : planifier la transition
L11	Gouvernance de l'Implémentation (Phase G) : assurer la conformité
L12	Durabilité numérique et éco-conception
L13	Le leadership politique de l'architecte
L14	Les patterns d'architecture du XXIe siècle

Dépasser le mythe technique

Architecture d'Entreprise

I — LE CAP

« The process of translating business vision and strategy into effective enterprise change by creating, communicating, and improving the key principles and models that describe the enterprise's future state and enable its evolution. » — Gartner, *Clarifies the Definition of the Term "Enterprise Architecture"*, 2008

« On n'apprend pas à faire de l'architecture de systèmes en lisant un livre, mais en pratiquant. » — Philippe Desfray et Gilbert Raymond, *TOGAF en pratique — Modèles pour l'architecture d'entreprise*, Dunod

Vidéo d'introduction (70s)

 Télécharger cette leçon en PDF

Le récif de Mutuelle Plus

Au début des années 2010, Mutuelle Plus fait figure de belle endormie. Fondée en 1961 par des fonctionnaires territoriaux pour garantir la santé de leurs collègues, l'organisation a grandi pendant cinq décennies jusqu'à protéger 900 000 personnes, soutenue par 800 collaborateurs. Dixième mutuelle de France, elle devrait dormir tranquille. Elle ne le peut plus.

La législation a changé. Les grandes compagnies d'assurance ont obtenu le droit de proposer des contrats santé, et la concurrence est devenue féroce. À l'intérieur, les signaux d'alerte se multiplient. Le portefeuille clients vieillit. Les coûts de couverture s'envolent. De gros contrats administratifs, perdus un à un, partent chez les concurrents qui montent en gamme. La direction technique veut réagir : « *Modernisons le SI, lançons un projet d'architecture.* »

L'inventaire est rude. Le catalogue de produits est devenu illisible tant il s'est étoffé. Plusieurs agences gèrent encore leurs clients sur des tableurs Excel locaux. Les contrats ne sont pas numérisés, et le délai de traitement d'un dossier dépasse les 72 heures. Le progiciel métier *Gest+*, déployé dix ans plus tôt comme une solution miracle, est devenu une camisole : chacun de ses écrans renvoie une erreur quand un cas déborde du cas standard, et les remboursements rendus à contretemps finissent par se retrouver dans la presse régionale.

Réunis en comité, les dirigeants poussent une conclusion convenue : *« lançons un vaste projet de refonte du SI ». * C'est une décision logique, presque inévitable. Et c'est, comme on va le voir, une erreur de catégorie — l'erreur-type que ce cursus entend vous apprendre à déjouer.

Une consultante externe est appelée. Elle ne commence pas par un audit technique. Elle organise **trente ateliers de travail** impliquant la majorité des employés — pas seulement la DSI. Les questions qu'elle pose ne portent pas sur les serveurs ni sur les bases de données. Elles portent sur les processus : « *Quand un client appelle pour résilier, que se passe-t-il ? Par qui ? Combien de temps ? Pourquoi ?* » Et petit à petit, sur ces trente ateliers, **dix grands projets de transformation** apparaissent — pas un, pas trois, dix. La mise en place d'un CRM, oui. Mais aussi la simplification de l'offre, l'amélioration de la qualité des données, la conformité RGPD, et cinq autres chantiers que personne n'aurait imaginés en ne regardant que le SI.

Le directoire, tenu au courant, fait quelque chose d'inhabituel : il **reaffirme le positionnement** de la mutuelle sur son cœur de métier, refuse de se disperser, et adopte une démarche d'architecture d'entreprise formalisée. Un *Corporate*

Balanced Scorecard est mis en place. Le cadre de gouvernance COBIT est mobilisé. Et pour la première fois, on cesse de gérer la DSI comme une tour d'ivoire repliée sur sa dette technique.

Mutuelle Plus n'est pas devenue une autre entreprise. Elle n'a pas racheté d'assureur, n'a pas changé de nom, n'a pas multiplié les partenaires. Elle a changé de **posture**. Elle a cessé de croire qu'une transformation réussie consistait à installer de nouveaux outils, et accepté l'idée qu'elle exigeait de repenser les processus, la gouvernance des données, et la place du SI dans la stratégie.

La leçon de cette histoire n'est pas « il faut faire trente ateliers ». Elle est plus simple, et elle est la première de ce cursus :

L'architecture d'entreprise est un acte de communication. Le SI n'est pas l'objet de la transformation ; il en est l'un des instruments. L'objet, c'est l'organisation qui doit servir ses clients.

C'est à cette condition qu'on traverse les brouillards.

Le récit de Mutuelle Plus finit en point d'interrogation plutôt qu'en résolution complète. C'est précisément ce qui en fait un récit utile pour une leçon d'architecture d'entreprise. La transformation n'est pas terminée, ne sera jamais terminée — et c'est ce qu'il faut comprendre dès la première leçon.

La leçon qui suit (L2) vous apprendra à poser le cadrage qui rend possible ce type de transformation. Mais d'abord, regardons ce qu'enseigne le récit de Mutuelle Plus à travers un schéma qui résume le cœur du message :

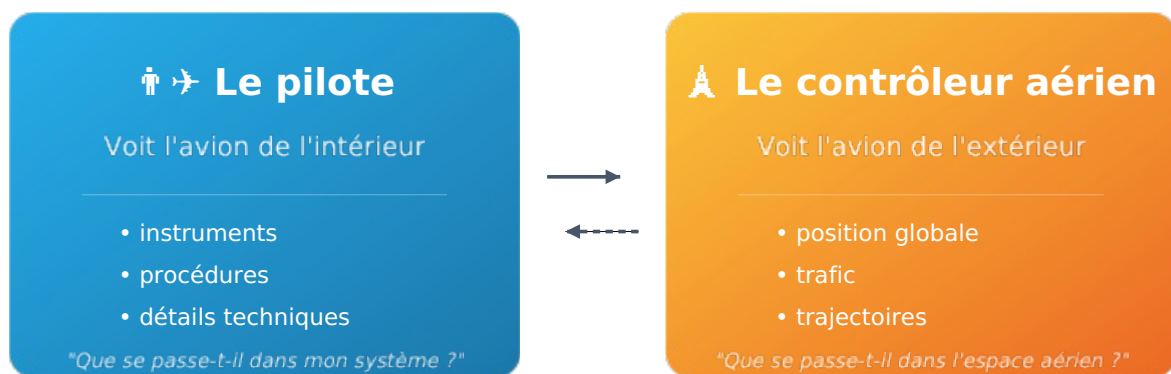


Schéma

L'analogie du pilote et du contrôleur aérien

Un même événement, deux perspectives légitimes. L'architecte traduit entre les deux.

même événement • deux vues légitimes



→ l'architecte traduit entre les deux perspectives

Premier principe — L'AE est un acte de communication

The Open Group ne définit pas l'architecture d'entreprise comme un exercice de modélisation technique. Elle la définit comme un processus de traduction — la traduction d'une vision et d'une stratégie en changements effectifs (TOGAF, Introduction and Core Concepts, Section 3.2). L'AE sert avant tout à rendre visible ce qui, sans elle, resterait dans la tête

de quelques-uns.

Cette définition a une conséquence pratique immédiate. Si votre interlocuteur ne peut pas comprendre votre schéma, votre schéma ne vaut rien — aussi élégant soit-il. « *L'architecture d'entreprise est avant tout un acte de communication entre direction générale, directions métiers et informaticiens* », écrivent Desfray et Raymond dans la préface de *TOGAF en pratique*. L'erreur du débutant consiste à dessiner des diagrammes pour lui-même : exhaustifs, jargonnants, illisibles. Le diagramme juste est celui qui **filtre l'information pour la partie prenante qui le consomme**.

L'analogie classique de TOGAF — celle du pilote et du contrôleur aérien — l'illustre parfaitement. Un pilote regarde un cockpit plein d'instruments ; un contrôleur aérien regarde un radar plein de plots. Les deux ont raison : ce sont deux points de vue sur le même espace aérien. L'architecte d'entreprise doit savoir produire **l'un ou l'autre** selon la salle où il parle. Devant un comité exécutif, il réduit le schéma aux grandes chaînes de valeur et aux capacités. Devant les experts FICO (Finance, Contrôle), il détaille les processus. Devant les développeurs, il montre les interfaces et les contrats de service. Devant les opérationnels, il résume les règles métier et les exigences non fonctionnelles.

La contrefaçon à éviter : l'« *implicit architecture* » — ces projets qui *prétendent* avoir une architecture mais ne peuvent fournir aucun diagramme de point de vue, aucune traçabilité des exigences, aucun principe directeur validé par un sponsor. C'est l'architecture qui vit dans la tête d'un seul expert, qui devient inutilisable le jour où il part, et qui n'a, en attendant, jamais instruit personne.

Et toi ? Regarde ton dernier livrable d'architecture. À qui était-il réellement destiné ? À toi-même pour clarifier ta pensée, ou à tes interlocuteurs pour les aider à décider ? Et qui n'a pas compris ton dernier schéma, alors qu'il aurait dû ?

Deuxième principe — La méthode structure l'exploration et prévient l'impasse

Sans méthode, l'architecture d'entreprise se transforme en expédition au petit bonheur. La méthode ADM (Architecture Development Method), cœur du framework TOGAF, structure rigoureusement la transformation de l'entreprise en huit phases (de A à H) articulées autour d'une gestion continue des exigences (TOGAF, Architecture Development Method, Ch. 2).

L'ADM fait trois choses que l'instinct ne fait pas. Premièrement, elle impose un **ordre** : avant de modéliser la cible, on s'accorde sur la vision ; avant de modéliser les applications, on valide les processus ; avant de planifier la transition, on a les écarts en main. Deuxièmement, elle rend **tracables** les décisions : chaque exigence métier peut être reliée à un composant d'architecture, et inversement. Troisièmement, elle **ferme les portes de sortie** qui mènent aux impasses — d'où la métaphore de la « checklist » chère à Desfray et Raymond : « *une checklist pour vérifier que la démarche d'architecture d'entreprise que vous êtes en train d'effectuer ne fait pas des impasses* ».

L'ADM ne prétend pas tout résoudre en un cycle. Un cycle ADM complet dure *réalistement* entre **six mois et deux ans** selon l'ampleur de la transformation et la maturité de l'organisation (TOGAF, *Introduction*, p. 6). C'est lent, et c'est précisément cette lenteur qui rend les transformations robustes.

L'ADM n'est pas non plus une prison. « *TOGAF fournit [...] un toolkit — il ne s'agit pas de mettre en œuvre 100 % des bonnes pratiques mais d'importer celles qui consolident les faiblesses de vos propres pratiques* » (Desfray et Raymond). On n'applique pas TOGAF comme on récite un catéchisme. On choisit, dans la boîte à outils, ce qui répond aux faiblesses observées, et on laisse le reste. Adapter la méthode s'appelle le *tailoring* — et c'est une compétence d'architecte, pas un écart de conduite.

La contrefaçon à éviter : l'« *over-modeling* » — ces projets où, faute de méthode, on se jette dans une modélisation exhaustive et ultra-détaillée de l'existant, pour produire des diagrammes illisibles pour les métiers, coûteux à maintenir, et obsolètes avant d'avoir été terminés. La parade est *top-down* d'abord, *bottom-up* ensuite, sur un périmètre ciblé.

Et toi ? Dans ton dernier chantier, as-tu suivi une méthode explicite (ADM ou autre), ou as-tu improvisé ?

Troisième principe — Le piège mortel : l'architecture après décision

Parmi tous les pièges que recense le corpus, le plus dangereux pour un débutant s'appelle l'« architecture après décision » (architecting after decision). Il tue la fonction d'AE plus sûrement que n'importe quel autre.

Le mécanisme est simple. L'architecte d'entreprise attend passivement d'être invité aux processus de planification stratégique ou budgétaire. Les choix technologiques ou d'investissements majeurs finissent par être actés par les dirigeants d'après les informations opérationnelles qu'ils ont sous la main le jour J. L'architecte est alors sollicité *après coup* pour documenter ou modéliser ce choix. Au mieux, son travail ne sert à rien — « *pointless* », disent les auteurs du corpus. Au pire, il démontre techniquement que la décision des dirigeants était mauvaise, ce qui génère de vives tensions et détruit la crédibilité de l'équipe d'AE pour longtemps.

L'histoire de Mutuelle Plus illustre exactement la parade. Le directoire, en refusant de lancer un « projet de refonte du SI » isolé, a évité l'« architecture après décision » en replaçant la transformation dans son cycle stratégique.

La parade : aligner impérativement les cycles de livraison d'architecture — notamment la Phase A (Vision) et les roadmaps stratégiques — sur le rythme et le calendrier budgétaire ou de planification de l'entreprise (*business cycle*). L'objectif n'est pas de tout livrer avant la décision, mais de fournir des **recommandations prêtes à l'emploi** (*decision-ready*) avant que la décision ne soit prise.

Deuxième contrefaçon à éviter : la confusion entre le référentiel d'AE et une CMDB. L'AE décrit les composants logiques d'architecture ; la CMDB gère les configurations physiques des serveurs. Confondre les deux, c'est condamner l'équipe d'AE à de la saisie de données opérationnelles, au lieu de conception stratégique.

***Et toi ?** Dans ton organisation actuelle, à quelle fréquence es-tu consulté sur les décisions stratégiques — une fois par an, jamais, ou après les décisions ? Quand pourrais-tu être consulté avant, et qu'est-ce qui t'en empêche ?*

Quatrième principe — La posture de l'architecte : praticien, communicant, humble

Le corpus insiste sur un point qui manque dans la plupart des manuels : l'architecture d'entreprise est un art difficile où l'expérience reste irremplaçable, où la lecture seule des théories ne suffit pas à faire un expert.

Trois attitudes résument la posture attendue.

Le praticien. « On n'apprend pas à faire de l'architecture de systèmes en lisant un livre, mais en pratiquant » (Desfray et Raymond). Ce cursus vous donne un cadre, des concepts, un vocabulaire, des outils. Aucun de ces éléments ne vous rend architecte. Ce qui vous rend architecte, c'est ce que vous en faites dans vos projets. La règle empirique du corpus est sans appel : *ne jamais concevoir un composant pour la réutilisation à moins d'avoir immédiatement au moins deux cas d'usage réels et distincts qui vont l'implémenter à court terme* (Desfray et Raymond). L'anticipation spéculative est une perte de temps.

Le communicant. Mutuelle Plus a réussi parce que la consultante a pris le temps de parler aux employés, pas seulement aux serveurs. Vous passerez 30 à 50 % de votre temps à expliquer, à écouter, à reformuler. Si vous n'aimez pas cette part du métier, l'AE n'est pas pour vous.

L'humble. Vous serez presque toujours le moins senior dans la salle quand vous présenterez à un comité exécutif. Vous n'avez pas la légitimité du poids hiérarchique ; vous avez celle de la clarté de vos modèles et de la solidité de vos arguments. Assumez votre ignorance des détails business ; assumez aussi que vos interlocuteurs détiennent des éléments que vous n'avez pas. La fonction d'AE est **une fonction de traduction**, pas de surplomb.

|> **Et toi ?** Sur tes trois derniers mois de pratique, quelle part de ton temps as-tu consacrée à lire / modéliser, et quelle part à écouter / expliquer ? Cette proportion est-elle soutenable pour le travail qui t'attend ?

Module complémentaire — Au-delà de la posture : le leadership politique de l'AE

Le quatrième principe parle d'attitudes (praticien, communicant, humble). Ce module ajoute une **dimension supplémentaire** : la dimension **politique**. Une posture d'humilité mal comprise peut devenir une posture de retrait ; une posture de communicant sans stratégie politique peut devenir une posture de transparence naïve. Ce module trace la frontière.

L'AE n'a pas d'autorité hiérarchique. C'est un fait structurel, pas un défaut. Le pouvoir formel (direction de projet, budget, sanction) appartient à d'autres : DSI, directeurs métiers, chefs de projet, RSSI. Le rôle de l'AE est de **rendre visibles** des choses que personne d'autre ne voit (cohérence transverse, dette technique, risques de solvabilité) et de **convaincre** que la cible technique sert une cible métier. Convaincre n'est pas imposer.

Cinq leviers d'influence sont à la disposition de l'architecte. Ils seront développés en détail dans la leçon 13 (L13 — Le leadership politique de l'architecte) ; cette section les introduit pour ancrer la posture dès la première leçon.

1. **L'expertise** : la légitimité technique est le ticket d'entrée. Sans elle, personne n'écoute. Mais elle ne suffit pas.
2. **La coalition** : on ne convainc pas seul. Identifier ses alliés naturels (RSSI, DAF, contrôle interne) et construire avec eux des arbitrages défendables.
3. **La persuasion par la valeur métier** : le jargon technique est un mur. Traduire en TCO, ROI, risque de solvabilité, time-to-market.
4. **L'échange et la réciprocité** : la transformation n'est pas un dû. Le *small win* (numériser un dossier critique pour désengorger un service) prouve la valeur de la coopération.
5. **La contrainte douce** : le comité d'architecture n'est pas un tribunal. C'est un lieu de **traçabilité** des écarts. La documentation des écarts est plus persuasive que l'interdiction.

Trois postures qui tuent la crédibilité (qui seront détaillées en L13) :

- **Donner des ordres** quand on n'a pas de pouvoir hiérarchique. C'est le meilleur moyen de créer des conflits de territoire.
- **Se cacher derrière les standards** (« la méthode l'impose »). Quand un interlocuteur entend cette phrase, il entend : « je ne sais pas justifier la valeur métier ».
- **Feindre de tout savoir** face à des experts qui ont vingt ans de maison. L'arrogance est suicidaire. Adopter la posture de l'**apprenant expert** : maîtriser la méthode et la vision, accepter d'apprendre les détails.

Cartographier le pouvoir avant de cartographier le SI. Cette phrase résume l'essentiel. Un schéma directeur qui ne commence pas par une cartographie des parties prenantes (Pouvoir × Intérêt — voir L2 Phase Préliminaire, module Mendelow) est un schéma qui s'effondrera au premier conflit politique.

La posture de l'architecte n'est pas une attitude statique. C'est un positionnement dynamique, ajusté en temps réel selon la salle, l'audience, l'enjeu. L'architecte qui ne bouge pas est un architecte qui finit par disparaître.

Application — Ma charte d'architecte

À rédiger séance tenante. Cinq éléments précis. À afficher à un endroit visible et à confier à deux personnes de confiance (mentor, pair ou manager).

1. **Mon manifeste personnel** — En une phrase, *pourquoi* je fais de l'architecture d'entreprise. Pas ce que je produis, mais ce que je veux produire *comme effet*.
2. **Mes trois personas prioritaires** — Les trois types d'interlocuteurs que je veux servir en priorité dans les 90 jours qui viennent (ex. : DSI, directeur métier, sponsor exécutif). Nommer les personnes réelles, pas les fonctions abstraites.
3. **Mes trois principes directeurs** — Trois règles opérationnelles que je m'engage à suivre (ex. : « toujours un diagramme par page de slide », « jamais plus de 9 objets par vue », « toute exigence métier reliée à un composant d'architecture »).
4. **Mes trois engagements datés** — Trois actions concrètes à des échéances précises (30, 60, 90 jours). Pas des vœux. Des livrables avec date, livrables et témoins.
5. **Mon premier allié** — Une personne à qui je confie ma charte dans les 7 jours, en lui demandant de me rappeler mes engagements à la fin de chaque période.

Cette page est votre premier livrable d'architecte. C'est le début de votre dossier.

Encadré Au lecteur / À l'animateur

Question d'ouverture (à poser à voix haute avant la lecture) : « Racontez-moi un projet de votre organisation où vous avez le sentiment qu'on a transformé le SI sans transformer l'organisation. Qu'est-ce qui a manqué ? »

Profils à observer chez l'apprenant :

- **Le technicien qui se cache derrière les diagrammes.** Produit des modèles impeccables et inutiles. *Question* : « À qui doit servir ce schéma ? Montrez-leur. »
- **L'architecte solo qui croit tout pouvoir porter.** Refuse de faire valider ses principes, livre seul, découvre trop tard. *Question* : « Qui vous a mandaté sur ce sujet ? »
- **Le pressé qui veut brûler les phases.** Saute directement aux applications. *Question* : « Que savez-vous du contexte métier de cette application ? »
- **Le dogmatique qui récite TOGAF.** Connaît le framework, ne l'adapte pas. *Question* : « Qu'avez-vous ajouté à TOGAF pour votre contexte ? »

Pièges d'animation :

- Ne pas laisser la discussion dériver vers la seule technique. Rappeler que la leçon 1 porte sur la **posture**, pas sur les outils.
- Ne pas laisser un apprenant croire qu'il « sait ». La charte (application) le ramènera à sa propre capacité à se poser les bonnes questions.

Si profil d'apprenant ≠ novice : ouvrir la discussion sur ce qui, dans l'organisation actuelle de l'apprenant, fait obstacle aux trois promesses du cursus (sécuriser, langage commun, aligner). Adapter les exemples à son secteur.

Ce que la leçon 2 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- Un **mémo d'une page** sur le contexte de votre organisation (taille, secteur, gouvernance du SI actuelle, culture de la décision).
- Le nom d'un **sponsor exécutif** potentiel à qui présenter votre charte dans les 30 jours.
- Un **premier obstacle** identifié dans votre pratique, à formuler en deux phrases au plus.

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Pourquoi l'architecture d'entreprise n'est-elle pas uniquement une question d'outils ? Quel rôle le récit de Mutuelle Plus joue-t-il dans cette leçon ?

A2 Quel est le métier réel d'un architecte d'entreprise selon cette leçon ? En quoi diffère-t-il d'un projeteur ou d'un consultant ?

A3 Qu'est-ce que l'analogie du pilote et du contrôleur aérien ? En quoi permet-elle de penser la communication dans une grande organisation ?

A4 Que signifie concrètement « ne jamais apparaître comme celui qui bloque » ? Pourquoi est-ce un principe de posture, pas de carrière ?

A5 Pourquoi le métier d'architecte d'entreprise est-il difficile à pratiquer ? Quelles sont les trois causes structurelles mentionnées ?

A6 Quel est le piège mortel de l'« architecture après décision » ? Pourquoi est-il si coûteux ?

B — AUTO-EVALUATION

B1 Sur ton dernier projet d'architecture, quel a été ton livrable principal ? Combien de parties prenantes l'ont réellement consulté ?

B2 Combien de fois as-tu été appelé pour « documenter » une décision déjà prise ? Combien de fois as-tu accepté ce rôle ?

B3 Sur tes cinq dernières semaines de travail, à quelle fréquence as-tu été consulté *avant* la décision plutôt qu'*après* ?

B4 Sur ton prochain livrable d'architecture, combien de parties prenantes réelles (nommées, pas des fonctions) ont été consultées ?

B5 Sur tes cinq derniers projets, combien ont été abandonnés en cours de route après une décision politique ? As-tu accepté ce rôle ?

B6 Sur tes cinq derniers projets, combien de fois as-tu réussi à intervenir avant la décision structurante ? Combien après ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Diagramme produit pour une seule partie prenante réelle, pas pour toi. (Échéance : J+14.)

C2 Atelier d'écoute opérationnelle conduit sur un processus réel. (Échéance : J+21.)

C3 Diagramme produit pour une seule partie prenante réelle, pas pour toi. (Échéance : J+14.)

C4 Atelier d'écoute opérationnelle conduit sur un processus réel. (Échéance : J+21.)

C5 Engagement personnel n°1 à 30 jours vérifiable par un témoin externe (quoi, quand, comment). (Échéance : J+30.)

C6 Décision prise avec un sponsor *avant* qu'il ne tranche, sur un sujet où tu étais jusque-là consulté *après*. (Échéance : J+45.)

La Phase Préliminaire : cadrer l'espace de jeu

Architecture d'Entreprise

I — LE CAP

« This Preliminary phase is about defining 'where, what, why, who, and how we do architecture' in the enterprise concerned. » — The Open Group, The TOGAF® Standard — Architecture Development Method, Phase Préliminaire

« Le diplôme ouvre des portes ; le caractère décide de ce que l'on devient une fois la porte franchie. »

Vidéo d'introduction (1 min 50)

 Télécharger cette leçon en PDF

Le bureau vide

Au début de l'année 2023, Élise se voit confier une mission qu'elle n'a jamais demandée. Son DSI, fraîchement nommé, l'a recrutée pour « monter une pratique d'architecture d'entreprise ». Le contexte est favorable : la société a fusionné trois entités deux ans plus tôt, le SI est hétérogène, le COMEX veut reprendre la main, et le cabinet d'audit qui a livré le diagnostic a suggéré avec force qu'« un architecte » résoudrait le sujet. Élise arrive avec un CV propre, une formation TOGAF certifiante, et un contrat de mission : « structurer la pratique AE et produire un premier plan de transformation ».

Son premier matin, elle découvre le bureau. La pièce est vide. Pas de fichiers, pas de mandats antérieurs, pas de budget alloué, pas de référentiel existant, pas d'équipe. Elle a été nommée pour exercer une fonction que personne n'avait définie avant elle. Son interlocuteur principal, le DSI, lui dit en substance : « *Tu es libre, structure ça comme tu veux.* »

Liberté, en pratique, est un mot qui recouvre beaucoup de choses. Pour Élise, il recouvre surtout une absence totale de périmètre, de gouvernance et de légitimité. Pendant les six premières semaines, elle fait ce que ferait n'importe quel professionnel consciencieux face à un tel blanc : elle lit, elle assiste à des réunions pour observer, elle se présente à quelques interlocuteurs, elle rédige des notes. Elle ne produit aucun livrable architectural, parce qu'elle n'a pas encore le mandat de le faire. À la fin du deuxième mois, elle reçoit un mail du DSI qui demande « *où en est-on sur le plan de transformation* ». Élise doit répondre quelque chose.

L'erreur classique — l'erreur que cette leçon vous apprendra à éviter — consiste à se jeter directement sur le plan de transformation. À ouvrir le chapitre ADM Phase B, à dessiner des vues ArchiMate, à produire une cible ambitieuse qu'aucun acteur n'a demandée et qu'aucune gouvernance ne validera. Élise aurait pu le faire. Elle était tentée. Son instinct de pédagogue lui disait que la production rapide démontrerait sa valeur.

Elle ne l'a pas fait. Au lieu de ça, elle a passé les six semaines suivantes dans des conversations. Avec le DSI, pour cadrer son mandat et ses indicateurs de succès. Avec la direction financière, pour comprendre les cycles budgétaires et les contraintes d'investissement. Avec les propriétaires des trois entités fusionnées, pour cartographier les jeux d'alliances et les points de blocage. Avec la DRH, pour comprendre comment une équipe d'architecture se constituerait et se financerait. Avec l'audit interne, pour récupérer le rapport qui avait déclenché la mission. À la fin du troisième mois, elle avait un document d'une douzaine de pages — pas un plan de transformation, mais une **charte de gouvernance**

de la pratique AE. Elle l'a présentée au COMEX. Elle a été validée en 45 minutes.

C'est ce document — et pas un diagramme ArchiMate — qui a rendu possible tout ce qui a suivi. Sans ce cadrage initial, chaque livrable architectural qu'Élise aurait produit aurait flotté dans le vide, sans ancrage, sans protecteur, sans financeur. Au lieu de ça, à la fin de la première année, Élise avait une équipe de quatre architectes, un budget récurrent, un comité d'architecture mensuel, et trois chantiers de transformation engagés. La clé n'avait pas été son talent à dessiner des vues. La clé avait été sa capacité à **cadrer l'espace de jeu avant de jouer**.

C'est exactement ce que fait la Phase Préliminaire. Le schéma suivant résume les six questions auxquelles cette phase répond :

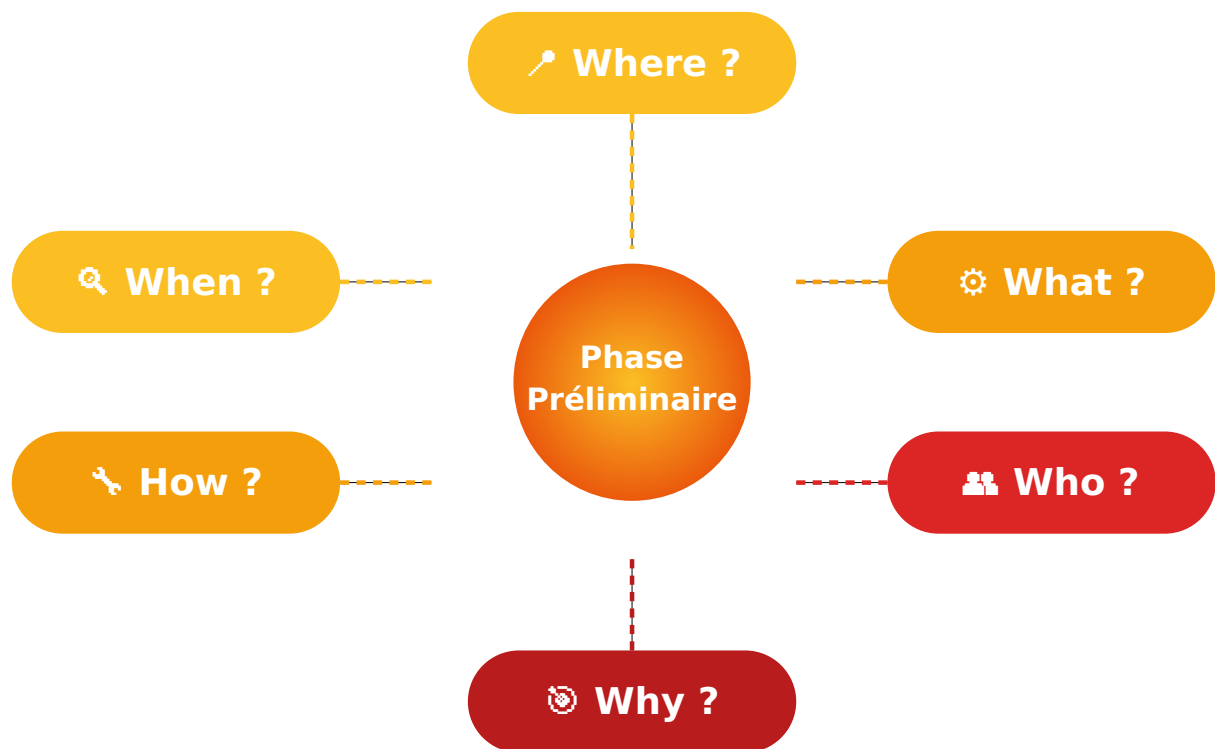
III

Schéma

Les 6 questions de la Phase Préliminaire

Le cadrage qui rend possible tout le reste

les 6 questions à poser avant tout



Premier principe — La Phase Préliminaire répond à cinq questions, pas à une seule

La Phase Préliminaire de l'ADM est souvent confondue avec une simple étape administrative. Elle est en réalité l'étape la plus structurante du cycle : elle répond à cinq questions interdépendantes qui définissent les conditions dans lesquelles tous les autres travaux deviendront possibles.

L'Open Group pose ces cinq questions explicitement (TOGAF, *Architecture Development Method*, Phase Préliminaire) :

- **Where** — Quel est le périmètre de notre pratique ? Sur quelle zone de l'organisation intervenons-nous, et avec quelle profondeur ?
- **What** — Quels livrables, quelles méthodologies, quels outils, quels standards utilisons-nous ?
- **Why** — Quelle est la valeur métier attendue ? Quels indicateurs de succès ?
- **Who** — Quels rôles, quelles responsabilités, quelle gouvernance, quelles instances de validation ?
- **How** — Quels processus internes, quelle cadence, quelle articulation avec les cycles budgétaires et stratégiques de l'organisation ?

Élise a passé un trimestre à cartographier ces cinq questions. Elle ne les a pas traitées dans l'ordre canonique — elle a commencé par le *why* (les indicateurs de succès que le DSI accepterait de défendre devant le COMEX), puis le *who* (qui validerait quoi), puis le *where* (sur quel périmètre on attaquait en premier), puis le *how* (cadence des comités), puis le *what* (standards et outils). Mais chacune des cinq réponses conditionnait les autres.

La contrefaçon à éviter est l'inverse exact de cette démarche : commencer par le *what* (les outils, les frameworks, les templates) avant d'avoir posé les autres questions. C'est l'erreur du débutant qui sort de sa certification TOGAF avec des diagrammes ArchiMate prêts à l'emploi et qui cherche où les brancher.

***Et toi ?** Sur les cinq questions de la Phase Préliminaire, à laquelle as-tu déjà répondu dans ton contexte actuel ? Et à laquelle te jettes-tu en premier quand tu n'as pas le temps ?*

Deuxième principe — Sanctuariser la frontière entre AE et CMDB avant le premier livrable

La confusion entre le référentiel d'Architecture d'Entreprise et une CMDB est l'un des pièges mortels recensés par le corpus (Desfray et Raymond, TOGAF en pratique). Cette confusion coûte cher quand elle n'est pas tranchée à la Phase Préliminaire.

Le mécanisme est connu. L'équipe AE hérite d'un référentiel vide (ou d'un outil déjà utilisé comme CMDB par les opérationnels). Pour produire ses premiers livrables rapidement, elle commence à saisir dans l'outil AE des composants physiques — versions de patchs, numéros de série, configurations serveurs — pour donner l'impression d'un référentiel garni. Très vite, l'équipe passe 70 % de son temps à maintenir des données opérationnelles qui ne sont pas son cœur de métier. Elle devient une équipe CMDB-bis qui n'a pas les moyens de tenir le rythme des opérationnels. Elle perd sa crédibilité sur les sujets architecturaux.

Élise a évité ce piège dès la Phase Préliminaire. Elle a explicitement écrit dans sa charte que le référentiel AE « hébergera les composants logiques d'architecture — *Architecture Building Blocks* (ABB) et modèles cibles/de transition — et ne contiendra pas de données d'exploitation ». Elle a prévu une interface de synchronisation avec la CMDB pour les indicateurs clés, mais pas de saisie manuelle redondante. Cette phrase dans la charte a évité dix réunions et trois ruptures de gouvernance.

La parade est simple à énoncer, difficile à tenir : **la frontière doit être explicite, écrite, et défendue**. Si elle ne l'est pas, chaque demande opérationnelle trouvera un chemin pour s'y glisser.

***Et toi ?** Dans ton organisation, qui tient aujourd'hui la CMDB ? Et qui est responsable du référentiel d'architecture ? Sont-ils la même équipe, deux équipes, personne ? Cette confusion est-elle tranchée à la Phase Préliminaire, ou latente ?*

Troisième principe — Identifier les parties prenantes hostiles

avant qu’elles ne bloquent

L’un des angles morts les plus coûteux de la Phase Préliminaire consiste à se concentrer sur les alliés et à négliger les opposants. C’est une erreur politique aussi bien que méthodologique.

Élise a identifié trois catégories de parties prenantes lors de sa cartographie initiale. D’abord, les alliés naturels — le DSI, l’audit interne, certaines directions métiers en demande. Ensuite, les neutres — la majorité des opérationnels, qui ne savent pas encore ce que la pratique AE va leur apporter. Enfin, les hostiles — et c’est là qu’Élise a passé du temps. Trois profils hostiles ont émergé :

- **L’architecte historique de la DSI**, qui exerçait cette fonction depuis dix ans sans cadre formel, et qui considérait l’arrivée d’Élise comme une remise en cause implicite de son travail.
- **Le responsable d’une entité fusionnée**, qui voyait dans la pratique AE un outil de recentrage du pouvoir sur la holding.
- **Le sponsor du progiciel X**, qui craignait qu’une pratique AE mature rende visible les limites de l’investissement déjà engagé.

Élise n’a pas évité ces opposants. Elle les a rencontrés. Elle a écouté ce qu’ils défendaient. Elle a inclus certains de leurs sujets dans la première vague de chantiers (ce qui les a transformés en alliés partiels). Elle en a neutralisé d’autres en obtenant du DSI des règles de gouvernance claires sur les cas-limites. Elle n’a pas ignoré l’architecte historique — elle l’a positionné comme référent sur l’un des domaines où il était légitime, et il est devenu un allié solide après quelques mois.

La contrefaçon à éviter est le déni : croire que les opposants vont se ranger une fois que les premiers livrables auront démontré la valeur de la pratique. Certains le feront. D’autres non. Mieux vaut les cartographier et les inclure dans le périmètre dès la Phase Préliminaire.

Module complémentaire — Cartographier avec Mendelow : la matrice Pouvoir / Intérêt

Identifier les parties prenantes *hostiles* est un bon début. Mais pour piloter efficacement la cartographie politique de l’AE, il faut passer d’une typologie qualitative (allié / neutre / hostile) à une **matrice tactique** qui croise deux dimensions : le **pouvoir** (formel et informel) et l’**intérêt** (vis-à-vis de l’architecture).

Le modèle classique d’**Andrew Mendelow** (1991) propose quatre quadrants :

Quadrant	Pouvoir	Intérêt	Tactique
Gérer activement	Élevé	Élevé	Co-construction. Intégrer ces acteurs dans le processus fractal pour qu’ils s’approprient les Building Blocks.
Satisfaire	Élevé	Faible	Rassurance et veto. Fournir des indicateurs de risque clairs pour maintenir leur soutien passif.
Tenir informé	Faible	Élevé	Respect de l’expertise. Valoriser leur mémoire institutionnelle et vendre la réduction de leur dette technique.
Surveiller	Faible	Faible	Communication minimale. Information descendante ciblée pour éviter de nourrir l’inquiétude.

Application à la cartographie Élise. Reprenons les trois profils hostiles identifiés :

- **L’architecte historique de la DSI** : pouvoir *informel* élevé (il connaît tout le SI), intérêt élevé (sa légitimité est en jeu). → Quadrant « Gérer activement ». Tactique : le positionner comme référent d’un domaine où il est légitime (stratégie d’inclusion transformatrice).

- **Le responsable d'entité fusionnée** : pouvoir formel élevé (direction d'une entité), intérêt faible (il défend son pré carré, pas l'AE elle-même). → Quadrant « Satisfaire ». Tactique : inclure dans la gouvernance un mécanisme qui **protège son autonomie** sur les sujets non critiques.
- **Le sponsor du progiciel X** : pouvoir élevé (il a investi dans le progiciel), intérêt élevé (son ROI dépend du succès de l'AE — qui doit montrer que le progiciel fonctionne). → Quadrant « Gérer activement ». Tactique : démontrer que l'AE **renforce** la valeur du progiciel plutôt qu'elle ne la menace.

Trois erreurs de cartographie à éviter :

1. **Sous-estimer le pouvoir informel.** Un lead dev respecté par 30 personnes pèse parfois plus qu'un chef de service isolé dans son organigramme. Cartographier le formel ET l'informel.
2. **Confondre intérêt et engagement.** Une partie prenante peut avoir un *intérêt* élevé pour l'AE sans *s'engager* dans sa réussite. Distinguer les deux.
3. **Cartographier une fois pour toutes.** La matrice est révisée à **chaque phase de l'ADM** : un acteur « faible » peut devenir « clé » si la transformation le concerne directement (ex. : un Responsable Métier devient critique au moment de la Phase B).

L'artefact produit. Cette cartographie alimente directement la **charte de gouvernance** (cinquième principe de cette leçon) : elle détermine qui signe quoi, qui est consulté, qui est informé, qui est laissé en surveillance. Sans elle, la charte reste théorique.

Le passage de la typologie « allié/hostile » à la matrice de Mendelow est un saut de maturité politique de l'AE. Il marque le moment où l'architecte passe d'une posture défensive (« qui va me bloquer ? ») à une posture tactique (« comment mobiliser chacun selon son quadrant ? »).

Et toi ? Sur ton périmètre AE actuel, qui sont les alliés naturels, les neutres, et les hostiles ? As-tu identifié les hostiles nommément, avec leurs arguments, et une stratégie d'inclusion ou de neutralisation ?

Quatrième principe — Aligner la Phase Préliminaire avec le cycle budgétaire de l'organisation

La Phase Préliminaire n'existe pas dans le vide organisationnel. Elle s'inscrit dans un calendrier qui n'est pas le vôtre — celui des décisions budgétaires, des plans stratégiques, des cycles COMEX.

Élise a eu la chance d'arriver six mois avant le cycle budgétaire annuel. Elle a utilisé ce délai pour faire valider sa charte par le COMEX en début de cycle, et pour présenter un premier plan de chantiers au moment où les directions métiers commençaient à préparer leurs demandes d'investissement. Sans ce synchronisme, ses propositions auraient été vues comme des coûts supplémentaires dans un budget déjà contraint.

L'alignement avec le cycle budgétaire n'est pas un détail tactique. C'est un élément structurant de la Phase Préliminaire. Une pratique AE qui démarre en décalage avec le calendrier des décisions budgétaires dépense son énergie à produire des livrables que personne ne peut financer. Une pratique AE qui démarre en synchronisme avec ce calendrier est en mesure de démontrer sa valeur dès le premier exercice.

La parade, ici, est d'**auditer le calendrier de l'organisation** avant de fixer le vôtre. Quand Élise a commencé sa mission, son premier livrable — avant même la charte — a été un document d'une page intitulé « Calendrier stratégique 2023 » qui recensait les dates butoirs du COMEX, du cycle budgétaire, des revues de plan stratégique, des fenêtres d'audit, des périodes de gel budgétaire. Ce document est devenu le squelette de sa Phase Préliminaire.

Et toi ? Quand démarre le prochain cycle budgétaire dans ton organisation ? À quand remonte le précédent ? Y a-t-il un gel budgétaire prévisible qui rendrait impossible la mise en place de ta pratique AE sur certaines fenêtres de l'année ?

Cinquième principe — La charte de gouvernance vaut plus que les premiers livrables

Le livrable central de la Phase Préliminaire n'est pas un diagramme. C'est un document d'une douzaine à une trentaine de pages, validé par l'instance décisionnelle supérieure, qui pose les cinq questions et leurs réponses.

Ce document — qu'on peut appeler charte de gouvernance, document de cadrage, ou *Statement of Architecture Work* dans la terminologie TOGAF — est ce qui rend tout le reste possible. Il est court (les chartes efficaces font 12 à 30 pages, pas plus). Il est validé au plus haut niveau (COMEX ou instance équivalente). Il est revu annuellement. Il contient au minimum :

1. **Périmètre** — zones couvertes, zones exclues, profondeur d'intervention.
2. **Standards** — frameworks adoptés (TOGAF, ArchiMate, DAMA DMBOK, etc.) et niveau de *tailoring*.
3. **Gouvernance** — instances, rôles, responsabilités, processus d'arbitrage.
4. **Indicateurs de succès** — comment la valeur de la pratique AE sera mesurée.
5. **Ressources** — équipe, budget, outils.
6. **Calendrier** — alignement avec les cycles stratégiques et budgétaires.

Élise a présenté sa charte au COMEX après trois mois de cartographie. La séance a duré 45 minutes. La charte a été validée. Trois ans plus tard, Élise considère que cette charte est son livrable le plus utile — devant les dizaines de modèles d'architecture qu'elle a produits depuis.

La contrefaçon à éviter : confondre la charte avec un livrable de modélisation. Une charte n'est pas un diagramme. Elle ne décrit pas l'architecture cible. Elle décrit les conditions dans lesquelles des architectures cibles pourront être conçues, validées et implémentées.

Et toi ? Si tu devais présenter ta charte de gouvernance au COMEX demain matin, l'as-tu déjà rédigée ? Sinon, combien de temps te faudrait-il pour la produire — en l'état actuel de ton cadrage ?

Application — Ma charte de gouvernance AE

À rédiger séance tenante. Sept éléments précis. À afficher à un endroit visible et à confier à deux personnes de confiance (mentor, pair ou manager).

1. **Mon périmètre** — La ou les zones de l'organisation où j'interviens en priorité. Les zones explicitement exclues. La profondeur d'intervention (cartographie simple, modèles cibles, transition).
2. **Mes standards adoptés** — Frameworks principaux (TOGAF, ArchiMate, DAMA, BPMN...). Niveau de *tailoring* appliqué ou prévu. Les standards que je refuse d'utiliser, et pourquoi.
3. **Ma gouvernance** — L'instance qui valide mes livrables. Mon sponsor direct. Le processus d'arbitrage en cas de désaccord. La fréquence des revues.
4. **Mes indicateurs de succès** — Trois à cinq indicateurs par lesquels le COMEX jugera ma pratique. À quoi je suis tenu, et à quoi je ne suis pas tenu.
5. **Mon équipe** — Les rôles que j'ai (ou n'ai pas). Les compétences que je cherche. Le budget dont je dispose.
6. **Mon calendrier** — L'alignement avec le prochain cycle budgétaire. Les fenêtres de gel à éviter. Les dates butoirs COMEX qui structurent mon année.

7. Mes parties prenantes hostiles — Les noms, leurs arguments, ma stratégie d'inclusion ou de neutralisation.

Cette charte est ton premier livrable de Phase Préliminaire. Elle est le socle de tout ce qui suivra.

Encadré Au lecteur / À l'animateur

Question d'ouverture (à poser à voix haute avant la lecture) : « Sur les cinq questions de la Phase Préliminaire, à laquelle as-tu déjà répondu dans ton organisation actuelle ? »

Profils à observer chez l'apprenant :

- **Le tacticien qui veut dessiner des diagrammes avant tout cadrage.** Démarre ses projets ArchiMate avant d'avoir validé son périmètre. *Question : « Qui a validé ton périmètre ? »*
- **L'opportuniste qui profite du flou pour élargir son périmètre.** Prend tout ce qui passe, sans cadrer ce qu'il rendra. *Question : « Quels livrables t'engages-tu à produire d'ici 90 jours ? »*
- **Le politique qui négocie la gouvernance au lieu de la poser.** Pèse chaque arbitrage, perd un temps infini, n'arrive jamais à un document validé. *Question : « Combien d'instances distinctes valides-tu actuellement un livrable ? »*
- **L'isolé qui croit qu'il peut mener la pratique seul.** Refuse de constituer une équipe, ne demande pas de budget, s'épuise. *Question : « Quel est ton prochain recrutement ? »*

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les détails des frameworks. Rappeler que la Phase Préliminaire est un acte de **cadrage organisationnel**, pas un exercice de modélisation.
- Ne pas confondre « cartographie des parties prenantes hostiles » avec « guerre politique ». Le but est d'**inclure et de neutraliser**, pas d'**exclure**.

Si profil d'apprenant ≠ praticien AE : ouvrir la discussion sur l'équivalent dans le rôle de l'apprenant. Toute fonction transverse (PMO, qualité, sécurité, data) traverse les mêmes questions de Phase Préliminaire. Adapter les exemples.

Ce que la leçon 3 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- Une **liste des frameworks** que vous utilisez réellement (TOGAF, ArchiMate, BPMN, autres) — ceux que vous avez *adoptés*, pas ceux que vous avez *lus*.
- Une **liste des instances de validation** que vous traversez actuellement pour faire valider un livrable architectural.
- Le **nom du sponsor** qui pourrait défendre votre pratique devant le COMEX dans les 30 jours.

Sources de cette leçon : TOGAF Standard 10^e (Phase Préliminaire) · Desfray & Raymond, *TOGAF en pratique* · Cas « Élise et le bureau vide » (anonymisé, inspiré de plusieurs missions réelles)

Leçon suivante : **L3 — Le Cycle ADM : le moteur méthodologique.**

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Quelles sont les cinq questions auxquelles la Phase Préliminaire doit répondre ? Pourquoi sont-elles toutes nécessaires ?

A2 Qu'est-ce que la « sanctuarisation de la frontière entre AE et CMDB » ? Pourquoi est-ce un acte fondateur ?

A3 Pourquoi faut-il identifier les parties prenantes hostiles dès la Phase Préliminaire, avant qu'elles ne bloquent le projet ?

A4 Quel est le lien entre cycle budgétaire et Phase Préliminaire ? Pourquoi faut-il aligner les deux ?

A5 Quels sont les sept éléments d'une charte de gouvernance d'AE ? Pourquoi chacun est-il nécessaire ?

A6 Qu'est-ce qu'une « erreur de cadrage » en AE ? Pourquoi est-elle plus coûteuse qu'une erreur d'exécution ?

B — AUTO-EVALUATION

B1 Sur ton dernier projet, combien de ces cinq questions as-tu réellement posées avant de démarrer ?

B2 La frontière entre ton périmètre AE et le périmètre CMDB est-elle claire, contestée, ou inexistante ? Qui est chargé de la clarifier ?

B3 Sur ton dernier projet, combien de parties prenantes hostiles as-tu identifiées **avant** qu'elles ne bloquent ? **Après** ?

B4 Ton cycle budgétaire est-il aligné avec ton cadrage d'architecture, ou as-tu démarré en décalage ?

B5 Combien des sept éléments de la charte as-tu formalisés aujourd'hui ? Lesquels sont implicites ?

B6 Sur ton dernier projet d'architecture, as-tu identifié les erreurs de cadrage avant l'exécution ? Combien en as-tu corrigées ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Rédiger la section « Périmètre » de ma future charte (qui fait quoi, ce qui est dedans, ce qui est dehors). (Échéance : J+14.)

C2 Documenter explicitement la frontière entre AE et CMDB (qui gère quoi, qui arbitre les conflits). (Échéance : J+21.)

C3 Version brouillon de ta charte de gouvernance (les 7 éléments). (Échéance : J+21.)

C4 Rencontre avec ton sponsor direct pour cadrer ton mandat et ses indicateurs de succès. (Échéance : J+14.)

C5 Entretien formel avec la partie prenante la plus hostile, pour comprendre ses arguments. (Échéance : J+21.)

C6 Validation de ta charte par l'instance supérieure (COMEX ou équivalent). (Échéance : J+60.)

Le Cycle ADM : le moteur méthodologique

Architecture d'Entreprise

I — LE CAP

« TOGAF fournit [...] un 'toolkit' — il ne s'agit pas de mettre en œuvre 100 % des bonnes pratiques mais d'importer celles qui consolident les faiblesses de vos propres pratiques. » — Philippe Desfray et Gilbert Raymond, *TOGAF en pratique — Modèles pour l'architecture d'entreprise*, Dunod

« Pas d'excellence sans caractère. »

Vidéo d'introduction

 Télécharger cette leçon en PDF

L'architecture comme un tour de France

Imaginez qu'on vous confie la responsabilité de tracer un Tour de France. Pas une course improvisée au GPS, mais un Tour classique, étape par étape, avec ses cols, ses plaines, ses contre-la-montre, ses jours de repos. Les organisateurs du Tour le font depuis plus d'un siècle. Ils ont un plan, des étapes, des rôles. Chaque étape a un départ, une arrivée, des concurrents, un classement intermédiaire. Le Tour ne réussit pas parce qu'il improvise à chaque étape : il réussit parce que la structure générale existe avant la première pédalée.

L'Architecture Development Method (ADM) est exactement cela pour l'architecture d'entreprise. C'est un tour structuré, avec ses phases, ses livrables, ses points de contrôle. L'Open Group l'a défini formellement à partir des pratiques de ses membres — grands comptes internationaux qui ont partagé ce qui marchait et ce qui échouait. Ce n'est pas une théorie : c'est une méthode éprouvée par des centaines de cycles complets.

L'ADM n'est pas un *waterfall*. C'est un **cycle itératif**. Il y a des phases dans un ordre nominal, mais on peut revenir en arrière, décaler, exécuter en parallèle selon les besoins du contexte. Cette itérativité est sa force : elle permet d'absorber les changements, les surprises, les arbitrages politiques sans devoir tout recommencer. Elle est aussi sa complexité : mal comprise, elle devient une boucle sans fin où l'on n'arrive jamais à boucler un livrable.

L'ADM est composé d'une **Phase Préliminaire** (que nous avons traitée en L2) et de **huit phases principales** (de A à H), articulées autour d'une **gestion continue des exigences** qui irrigue l'ensemble. Chaque phase a des intrants, des activités, des livrables, des sorties. Chaque transition entre phases est un point de validation. Le cycle complet — Phase A → Phase H — dure en pratique entre **six mois et deux ans**, selon l'ampleur de la transformation et la maturité de l'organisation (TOGAF, *Introduction*, p. 6).

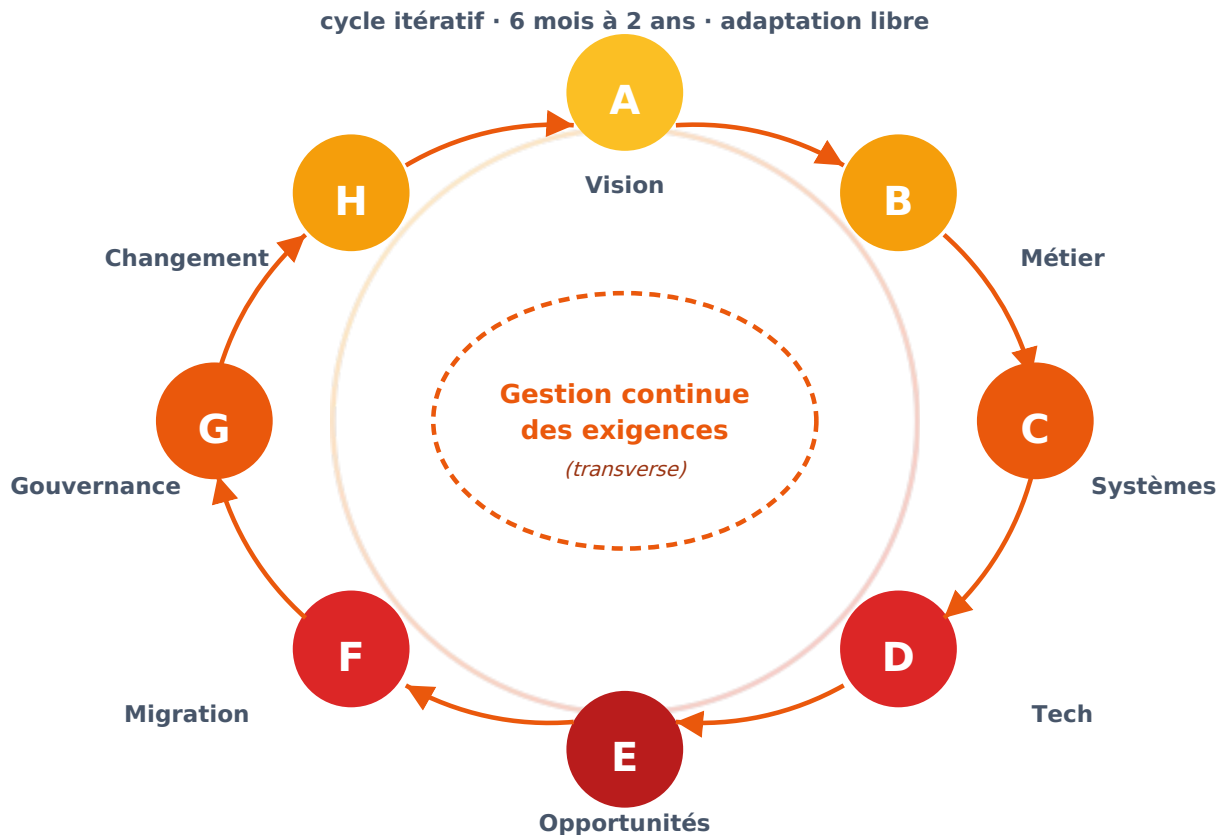
Le but de cette leçon n'est pas de mémoriser les noms des phases. Le but est de comprendre **pourquoi l'ADM fonctionne comme un cycle**, et **comment en tirer parti sans se laisser piéger par sa rigidité apparente**. Voici le cycle ADM complet :



Schéma

Le cycle ADM — Phases A à H + gestion continue

Itératif, pas séquentiel



Premier principe — L'ADM est un cycle, pas un waterfall

La confusion la plus coûteuse au sujet de l'ADM consiste à le lire comme un processus en cascade : on finit la phase A, on passe à B, puis C, etc. Ce n'est pas le cas. L'ADM est un cycle où chaque phase peut être ré-exécutée, raffinée, ajustée en fonction des découvertes faites en aval.

Le schéma canonique de l'ADM ressemble à un sablier ou à une roue : on entre par la Phase Préliminaire, on traverse A → B → C → D → E → F → G → H, et on revient en arrière pour un nouveau cycle. À chaque cycle, l'architecture devient plus précise, plus alignée, plus implémentable. Les exigences ne sont pas figées : elles sont gérées en continu, depuis la Phase A jusqu'à la Phase H, et continuent même au-delà.

Pourquoi cette structure cyclique ? Pour deux raisons que Desfray et Raymond soulignent dans *TOGAF en pratique*. Premièrement, l'architecture n'est pas un état final mais un **processus d'amélioration continue** : chaque cycle affine ce que le précédent a appris. Deuxièmement, les exigences métier évoluent pendant que l'architecture se construit : un nouveau directeur arrive, un concurrent sort un produit disruptif, une réglementation change. Un waterfall pur livrerait une architecture obsolète le jour de sa remise. Un cycle absorbe ces changements sans devoir tout reprendre.

L'itération n'est pas un désordre. Chaque cycle a un **périmètre** (un sujet précis qu'on traite dans le cycle courant) et un **degré d'engagement** (jusqu'à où on pousse les phases dans ce cycle). Un cycle peut couvrir toute l'organisation sur les phases A et B seulement (cadrage métier), puis un autre cycle peut approfondir la phase C sur un sous-domaine (architecture des données d'un processus spécifique). Les combinaisons sont infinies.

Et toi ? Quand tu mènes un projet d'architecture dans ton organisation, est-ce que tu boucles

explicitement un cycle complet avant d'enchaîner, ou est-ce que tu avances en rafales sans jamais fermer la boucle ? Quel est l'effet sur ta crédibilité quand le COMEX te demande « où en es-tu » ?

Deuxième principe — Les exigences sont au centre, pas en sortie

L'une des erreurs classiques consiste à traiter les exigences comme un livrable de début de projet — un document qu'on produit en Phase A et qu'on oublie ensuite. L'ADM pose au contraire que les exigences sont un objet vivant qui irrigue toutes les phases.

L'Open Group appelle cela la **gestion continue des exigences** (*Requirements Management*). Ce n'est pas une phase — c'est un processus transverse qui dure toute la durée du cycle. Chaque phase A → H produit ou raffine des exigences. Chaque revue de phase valide ou rejette des exigences. Chaque décision d'architecture est tracée jusqu'à une exigence explicite.

Pourquoi cette obsession de la traçabilité ? Parce qu'elle est ce qui distingue une architecture décisionnelle d'une architecture décorative. Une architecture décorative décrit un état futur idéal sans l'ancrer dans les exigences réelles du métier — elle reste sans défenseur quand le budget se serre. Une architecture décisionnelle relie chaque composant à une exigence nommée, à un sponsor identifié, à un indicateur mesurable. Quand le budget se serre, ce n'est pas l'architecture qui se contracte — c'est l'arbitrage qui devient possible : on coupe ce qui n'est adossé à aucune exigence vive, on protège ce qui est adossé à une exigence critique.

La contrefaçon à éviter est précisément l'inverse : une architecture où les exigences sont implicites, dispersées entre la tête des architectes et quelques slides éparses. C'est l'architecture qui flotte. C'est l'architecture qui meurt à la première vague.

***Et toi ?** Pour ton dernier livrable d'architecture, combien d'exigences sont explicitement tracées dans un document validé ? Et pour chaque composant du modèle cible, peux-tu remonter à l'exigence métier qui l'a justifié ?*

Troisième principe — Chaque phase a un livrable nommé, pas une étape “plus tard”

Le deuxième piège classique consiste à traiter les phases comme des étapes administratives dont les livrables peuvent être remis “plus tard”. L'ADM exige au contraire qu'à chaque phase corresponde un livrable nommé, validé, archivé.

Le schéma TOGAF liste pour chaque phase A → H les intrants (*Inputs*), les activités (*Steps*), les livrables (*Deliverables*), et les sorties (*Outputs*). Cette nomenclature n'est pas un détail bureaucratique. C'est ce qui rend possible la traçabilité, la gouvernance, et la capitalisation entre cycles. Sans livrables nommés, chaque cycle est une expérience à chaque fois nouvelle — et la pratique AE ne progresse pas.

L'illustration concrète est la Phase A (Vision). Le livrable central est le *Statement of Architecture Work* — un document d'une dizaine de pages qui pose le périmètre, les parties prenantes, les critères de succès, les contraintes, les hypothèses. Sans ce document, la Phase B (Architecture Métier) démarre dans le flou. Sans Phase B documentée, la Phase C (Architecture des Données) démarre dans le flou. Sans Phase C documentée, la Phase D (Architecture Applicative) démarre dans le flou. Et ainsi de suite, jusqu'à ce que l'équipe ne sache plus pourquoi elle produit ce qu'elle produit.

La parade n'est pas de produire tous les livrables en avance — c'est de les produire **au moment où la phase se**

termine, dans leur version minimale viable, validés par l'instance de gouvernance. Un *Statement of Architecture Work* de trois pages qui existe vaut mieux qu'un document de vingt pages qui n'existe pas.

Et toi ? Pour ton dernier cycle d'architecture, combien de phases ont produit leur livrable nommé en temps voulu ? Et combien ont glissé vers "on l'a fait à l'oral" ?

Quatrième principe — Le *tailoring* est une compétence, pas un écart

Le tailoring — l'adaptation de l'ADM au contexte — est la compétence la plus sous-estimée de la pratique AE. Bien appliqué, il distingue les architectes matures des techniciens qui récitent TOGAF comme un catéchisme.

L'ADM propose un cadre nominal. Mais chaque organisation a ses contraintes : taille, culture, cycle budgétaire, maturité numérique, profil de risque. Un cycle ADM qui s'exécute tel quel, sans adaptation, produit presque toujours un livrable inadapté au contexte. Un *tailoring* qui néglige les bonnes pratiques du cadre produit un livrable qui ne ressemble plus à de l'architecture d'entreprise.

La compétence de *tailoring* consiste à **identifier ce qui, dans le cadre, est essentiel à la discipline AE, et ce qui peut être adapté sans perte de substance**. Les phases A → H sont essentielles — elles structurent le cycle. La gestion des exigences est essentielle — elle garantit la traçabilité. Mais le nombre exact d'itérations par phase, le périmètre des livrables, le formalisme des revues, la cadence des comités : tout cela se négocie avec le contexte.

Desfray et Raymond résument la chose ainsi : « *TOGAF fournit un 'toolkit' — il ne s'agit pas de mettre en œuvre 100 % des bonnes pratiques mais d'importer celles qui consolident les faiblesses de vos propres pratiques* ». C'est la définition même du *tailoring* : un choix actif de ce qui consolide l'existant, pas une exécution servile du cadre.

La contrefaçon à éviter : le *tailoring* par ignorance — celui qui consiste à sauter des phases parce qu'on ne sait pas les exécuter, en prétendant que "c'est du *tailoring*". Ce n'est pas du *tailoring*. C'est de l'amateurisme qui se pare des atours de la méthode.

Et toi ? Sur ton dernier cycle ADM, quels éléments as-tu adaptés au contexte ? Et quels éléments as-tu sauté "parce qu'on n'avait pas le temps" ? La différence entre les deux est ta maturité de praticien.

Cinquième principe — Un cycle complet prend du temps, et c'est précisément ce qui le rend robuste

Le dernier piège à déjouer est celui de l'impatience : croire qu'on peut mener un cycle ADM complet en trois mois, parce que "le marché n'attend pas". C'est confondre vitesse et précipitation.

L'Open Group évalue la durée réaliste d'un cycle ADM complet entre **six mois et deux ans** (TOGAF, *Introduction*, p. 6). Ce n'est pas une fourchette théorique. C'est la médiane observée dans les organisations qui mènent des cycles aboutis. Pourquoi ? Parce qu'un cycle complet implique de traverser la Phase Préliminaire (cadrage), les phases A à H (production d'architectures As-Is, cible, de transition), les phases F et G (planification et gouvernance de l'implémentation), sans compter les itérations et les validations de gouvernance.

L'impatience produit deux types de dégâts. Premièrement, les **cycles avortés** — on déclare un cycle "complété" alors que seules les phases A et B ont vraiment été menées ; les phases C à H sont restées à l'état d'ébauche. Le résultat : une architecture cible sans implémentation, ou une implémentation sans architecture de transition. Deuxièmement, les **cycles compressés** — on tente de mener les phases en parallèle pour gagner du temps, ce qui produit des livrables

bâclés et des incohérences entre couches.

La parade n'est pas la lenteur par principe. C'est la **planification réaliste dès la Phase Préliminaire** : annoncer au sponsor que le cycle prendra X mois, négocier les jalons en conséquence, protéger les fenêtres de production contre les interruptions opérationnelles. C'est cette discipline de cadrage temporel qui distingue les architectes matures des pompiers permanents.

***Et toi ?** Sur ton dernier cycle annoncé à ton sponsor, quelle durée as-tu promise ? Et quelle durée a-t-il réellement fallu ? L'écart entre les deux est ton indicateur le plus fiable de crédibilité.*

Application — Ma roadmap ADM 90 jours

À rédiger séance à séance. Trois livrables précis, datés, à afficher et à confier à deux personnes de confiance.

1. **Mon périmètre de cycle** — Le sujet précis que je traite dans mon prochain cycle ADM. Ni trop large (toute l'organisation), ni trop étroit (un seul composant). Un périmètre qui peut être cadré en 90 jours.
2. **Mon cadrage temporel** — Pour chaque phase que je mène réellement (pas toutes les phases nominales), une durée estimée et une date butoir. Justification du *tailoring* appliqué.
3. **Mes trois exigences critiques** — Les trois exigences métier qui doivent absolument être couvertes par ce cycle. Avec leur sponsor nommé et leur indicateur de succès.

Ce document de trois pages est ton mon livrable de Phase A. C'est lui que tu présenteras au sponsor avant de commencer.

Encadré Au lecteur / À l'animateur

Question d'ouverture (à poser à voix haute avant la lecture) : « Combien de cycles ADM as-tu menés jusqu'à maintenant dans ton organisation ? Combien en as-tu bouclés ? »

Profils à observer chez l'apprenant :

- **Le récitant qui sait l'ADM par cœur mais ne l'a jamais pratiqué.** Énumère les phases, cite les livrables, n'a jamais vu un *Statement of Architecture Work* validé. *Question :* « Quel est ton dernier livrable de Phase A ? »
- **L' impatient qui confond vitesse et précipitation.** Promet des cycles de trois mois, livre des architectures sans implémentation. *Question :* « Combien de temps a duré ton dernier cycle complet ? »
- **Le rigoriste qui applique TOGAF sans *tailoring*.** Livre des livrables inadaptés au contexte parce qu'il a voulu suivre le cadre à la lettre. *Question :* « Quels éléments du cadre as-tu adaptés au contexte ? »
- **Le démineur qui improvise chaque cycle comme une urgence nouvelle.** N'a aucune capitalisation, refait les mêmes erreurs à chaque cycle. *Question :* « Quelle trace laisses-tu après chaque cycle ? »

Pièges d'animation :

- Ne pas laisser la discussion dériver vers la mécanique des phases (A, B, C...). Rappeler que la leçon porte sur **les principes qui sous-tendent le cycle**, pas sur le détail opérationnel.
- Ne pas confondre *tailoring* et paresse. Un *tailoring* se justifie, se documente, se défend. Une improvisation se constate, se subit, se regrette.

Si profil d'apprenant ≠ praticien AE : ouvrir sur l'équivalent dans le rôle de l'apprenant. Tout projet structurant

traverse des phases analogues — cadrage, conception, livraison, transition, gouvernance. Adapter les exemples.

Ce que la leçon 4 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- **Un périmètre précis** pour votre prochain cycle d'architecture ou de projet structurant.
- **Une durée estimée, jalonnée** par les phases que vous mènerez réellement.
- **Trois Exigences critiques** avec leur sponsor nommé et leur indicateur de succès.

Sources de cette leçon : TOGAF Standard 10^e (ADM, ADM Techniques) · Desfray & Raymond, *TOGAF en pratique*

Leçon suivante : **L4 — La Vision (Phase A) : éviter les choix hâtifs.**

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Qu'est-ce que l'ADM ? Quelles sont les huit phases principales ? Quel est le rôle de la gestion continue des exigences ?

A2 Pourquoi l'ADM est-il un cycle et non un waterfall ? Quel est l'intérêt de cette cyclicité ?

A3 Qu'est-ce que la « gestion continue des exigences » ? Pourquoi est-elle transverse plutôt qu'un livrable de phase ?

A4 Qu'est-ce que le « tailoring » dans TOGAF ? Quels sont les niveaux d'adaptation possibles ?

A5 Pourquoi le cycle ADM dure-t-il en pratique entre six mois et deux ans ? Quels facteurs font varier cette durée ?

A6 Quels sont les livrables canoniques de chaque phase ADM (A, B, C, D, E, F, G, H) ?

B — AUTO-EVALUATION

B1 Combien de phases d'un cycle ADM as-tu réellement traversées sur ton dernier projet ? Lesquelles ont été escamotées ?

B2 As-tu déjà pratiqué la gestion continue des exigences ? Combien d'exigences as-tu tracées sur ton dernier projet ?

B3 As-tu pratiqué le tailoring TOGAF, ou as-tu appliqué le cadre tel quel ? Quel a été l'effet ?

B4 Sur tes cinq derniers projets, combien ont duré plus de 12 mois ? Combien ont été interrompus avant la fin du cycle ?

B5 Combien de livrables TOGAF canoniques as-tu produits sur ton dernier cycle ? Lesquels as-tu improvisés ?

B6 Sur ton dernier cycle ADM, combien de tours complets as-tu réalisés avant de considérer l'architecture cible stabilisée ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Identifier le livrable canonique de chaque phase (A → H) de mon prochain cycle, et l'adapter à mon contexte. (Échéance : J+7.)

C2 Définir le rythme de mon cycle : revue trimestrielle, gouvernance mensuelle, ajustement annuel. (Échéance : J+14.)

C3 Identifier 3 exigences critiques pour ce cycle, avec leur sponsor nommé et leur indicateur de succès. (Échéance : J+21.)

C4 Présenter le livrable Phase A (3 pages) au sponsor et obtenir son accord avant de démarrer. (Échéance : J+30.)

C5 Documenter explicitement le *tailoring* appliqué (1 page) — ce qui est conservé tel quel, ce qui est adapté, ce qui est écarté. (Échéance : J+14.)

C6 Premier livrable de phase produit et validé par l'instance de gouvernance dans les 30 jours. (Échéance : J+30.)

La Vision (Phase A) : éviter les choix hâtifs

Architecture d'Entreprise

II — LA PROFONDEUR

« The Architecture Vision phase sets the scope, constraints, and expectations for a TOGAF architecture project. » — The Open Group, TOGAF Series Guide — Architecture Vision (G174e)

« Une décision sans architecture est une décision sans lendemain. »

Vidéo d'introduction

 Télécharger cette leçon en PDF

L'avion qui décolle sans plan de vol

Au printemps 2018, Bertrand prend la tête de la DSI de Discount Voyages — un voyageur en ligne de taille moyenne, 250 salariés, 80 millions d'euros de chiffre d'affaires. L'entreprise a doublé son volume de réservations en trois ans grâce à un site web performant, mais le système d'information est devenu un monstre spaghetti : douze applications métier qui ne se parlent pas, une base de données clients éclatée en trois silos, une équipe de développement qui travaille à 80 % sur la dette technique plutôt qu'à la création de valeur.

Le COMEX est clair : « *il faut moderniser le SI* ». La question est **comment**. Trois visions s'affrontent. Le directeur commercial veut un « *site web refondu avec un moteur de recommandation IA* » — il a vu un concurrent le faire. Le directeur financier veut un « *progiciel intégré qui remplace tout* » — il a vu trois cabinets lui promettre des ROI mirifiques. La directrice des opérations veut un « *outil de gestion des opérations back-office* » — elle étouffe dans les tableaux.

Les trois visions sont incompatibles. Elles coûteraient ensemble plus de douze millions d'euros. La trésorerie ne permet pas d'en financer plus de trois. Et personne n'a posé la question préalable : **que cherche-t-on vraiment à résoudre ?**

C'est exactement le rôle de la **Phase A — Architecture Vision**. Son but n'est pas de trancher entre les trois visions. Son but est de poser un cadrage qui rend la décision possible et défendable : périmètre, contraintes, parties prenantes, critères de succès, exigences principales. Une fois ce cadrage posé, les trois visions peuvent être comparées objectivement, sans tomber dans le piège du plus-fort-qui-parle-le-mieux.

L'Open Group l'a défini ainsi : « *The Architecture Vision phase sets the scope, constraints, and expectations for a TOGAF architecture project* ». C'est court. C'est exactement ce que Bertrand n'avait pas fait : poser le scope, les contraintes, et les attentes. Il avait foncé sur la première vision venue — celle du commercial, qui avait la voix la plus forte.

C'est ce que cette leçon cherche à vous apprendre : **comment poser une Phase A qui transforme une décision politique en une décision technique argumentée.**

Le schéma suivant résume la mécanique que cette leçon cherche à transmettre :

Premier principe — La Phase A est l'antidote à l'architecture après décision

Le piège de l'« architecture après décision » est le plus coûteux de tous. La Phase A est l'antidote, à condition d'être menée avant que les décisions structurantes ne soient prises.

L'« architecture après décision » consiste à être appelé après que la direction a déjà tranché (« on refait le site web », « on achète le progiciel X »), avec pour mission de produire les plans qui justifient la décision. Ce n'est plus de l'architecture, c'est de la documentation rétrograde. Le rôle de l'architecte est réduit à celui d'un projeteur qui dessine ce que le maître d'ouvrage a déjà décidé.

La Phase A remet l'architecte à sa place : **avant la décision**. Son livrable central n'est pas un modèle de cible ; c'est un **Statement of Architecture Work** qui pose les questions préalables. Ce document a une fonction politique autant que technique : il rend visible le périmètre des choix à faire, les parties prenantes à consulter, les critères de succès à respecter.

Chez Discount Voyages, Bertrand aurait pu tenir la Phase A en six semaines. Mais il faut, pour cela, qu'il accepte de ne pas livrer tout de suite. La tentation est grande de produire une cible pour faire plaisir au COMEX. Résister à cette tentation, c'est précisément ce qui distingue un architecte d'un projeteur.

La parade : accepter que la Phase A est un acte de **non-production** — on ne livre pas de cible, on livre une méthode pour en choisir une. C'est ce qui rend cette phase difficile à vendre aux directions habituées à des livrables tangibles.

***Et toi ?** Combien de fois as-tu été appelé pour « documenter » une décision déjà prise ? Combien de fois as-tu accepté ce rôle sans protester ? Qu'aurais-tu pu faire pour reprendre la main en amont ?*

Deuxième principe — Les Business Scenarios capturent les exigences par le dialogue

Une Phase A réussie ne commence pas par un diagramme. Elle commence par des conversations structurées avec les parties prenantes. Le cadre TOGAF appelle ces conversations les Business Scenarios.

L'Open Group a publié un guide dédié à cette technique : le *TOGAF Series Guide to Business Scenarios* (G176e). L'idée est de remplacer les discussions vagues (« il nous faut un nouveau SI ») par des dialogues structurés autour de scénarios concrets. Chaque un le suit une méthode en six étapes :

1. **Identifier le problème** — quel est le problème concret que les acteurs essaient de résoudre ?
2. **Identifier et modéliser les acteurs humains** — qui sont les parties prenantes, et quel est leur intérêt dans le problème ?
3. **Identifier et modéliser les acteurs informatiques** — quels systèmes existants jouent un rôle dans le problème ?
4. **Identifier et modéliser les intérêts des acteurs** — qu'est-ce qui compte pour chaque acteur ? Quels sont ses critères de succès ?
5. **Identifier et modéliser les dépendances** — quelles relations entre les acteurs ? Quels événements déclenchent quoi ?
6. **Vérifier et raffiner le un le** — confronter le un le aux parties prenantes, ajuster, valider.

L'avantage de cette méthode est qu'elle **transforme des opinions en énoncés testables**. Au lieu de dire « le commercial veut un nouveau site web », on peut dire « le commercial veut réduire le taux d'abandon de panier de 35 % à 20 % d'ici 18 mois, mesurable via Google Analytics ». L'exigence devient mesurable. Le débat devient technique, plus

politique.

Chez Discount Voyages, Bertrand a appliqué la méthode des Business Scenarios sur trois semaines. Il a rencontré le commercial, le financier, l'opérationnel, et trois clients clés. À l'issue, il avait trois scénarios formalisés, avec pour chacun un problème nommé, des acteurs identifiés, des intérêts mesurables, et des résultats attendus datés. C'est sur cette base que la Phase A a pu être écrite — et que le COMEX a pu trancher de manière informée.

***Et toi ?** Sur ton dernier projet, as-tu formalisé les exigences en scenarios structurés avant de produire le moindre schéma ? Ou as-tu directement dessiné des diagrammes sans avoir posé les questions préalables ?*

Troisième principe — Formuler des objectifs SMART, pas des vœux

Une Phase A qui contient des objectifs vagues produit des architectures vagues. La parade est la formulation SMART — pas pour cocher une case de consultant, mais pour rendre les exigences testables.

L'acronyme SMART (Spécifique, Mesurable, Atteignable, Réaliste, Temporellement défini) est un vieux réflexe du conseil en organisation. Il est souvent moqué pour son côté formulaire. Il reste utile en Phase A, à condition d'être pris au sérieux.

Un objectif **non-SMART** : « *moderniser le SI de Discount Voyages* ». Que signifie « moderniser » ? Comment mesure-t-on ? Pour quand quel cet objectif a-t-il un sens, il doit passer quatre tests :

- **Spécifique** — un énoncé clair que deux interlocuteurs comprendraient de la même manière
- **Mesurable** — un indicateur chiffré, et une source de mesure identifiée
- **Atteignable** — une trajectoire crédible, pas une ambition hors de portée
- **Réaliste** — un alignement avec les contraintes réelles (budget, équipe, temps)
- **Temporellement défini** — une date butoir, pas un horizon vague

Un objectif **SMART** : « *Réduire le taux d'abandon de panier de 35 % à 20 % d'ici 18 mois, mesurable via Google Analytics, sur la base d'une refonte du tunnel d'achat* ». La différence est considérable : le premier ne peut pas être contesté parce qu'il ne peut pas être vérifié ; le second peut l'être.

La contrefaçon à éviter : la formulation SMART purement cosmétique — des objectifs qui *paraissent* SMART mais qui ne sont en réalité pas mesurables (parce que la source de mesure n'existe pas, ou parce que l'indicateur ne reflète pas vraiment l'intention). La parade est de tester chaque indicateur en demandant « comment mesure-t-on cet objectif aujourd'hui, et qui a accès à cette mesure ? ». Si la réponse est floue, l'objectif n'est pas SMART en pratique.

***Et toi ?** Sur tes cinq derniers livrables d'architecture, combien d'objectifs sont SMART au sens propre ? Combien sont SMART en apparence mais invérifiables en pratique ?*

Quatrième principe — Le Diagramme de Concept de Solution est le livrable central

Une Phase A ne produit pas de modèle de cible détaillé. Elle produit un Diagramme de Concept de Solution — un schéma qui montre les grandes options architecturales à l'étude, sans prétendre à l'exhaustivité.

Le Diagramme de Concept de Solution est l'outil qui rend la Phase A concrète. C'est un schéma volontairement haut niveau — quelques boîtes, quelques flèches, des couleurs — qui représente les grandes options en lice. Sa fonction n'est pas de tout dire, mais de **rendre visible le périmètre de la décision**.

Pour Discount Voyages, Bertrand a produit un Diagramme de Concept de Solution qui présentait trois options :

- **Option A** — Refonte du site web avec moteur de recommandation IA
- **Option B** — Acquisition d'un progiciel intégré (avec migration complète des données)
- **Option C** — Outil back-office de gestion des opérations, sans toucher au front

Chaque option était représentée par un schéma simplifié, avec ses dépendances principales, ses risques, ses coûts estimés. Le COMEX a pu comparer les trois options en 45 minutes, avec un support visuel clair, sans se perdre dans les détails.

Le Diagramme de Concept de Solution n'est pas une cible — c'est un **menu**. Il sert à choisir, pas à exécuter. Une fois l'option choisie, le reste du cycle ADM (Phase B, C, D...) détaillera l'option retenue.

***Et toi ?** Sur ton dernier projet d'architecture, as-tu produit un schéma haut-niveau qui présentait les options en lice — ou as-tu foncé sur une seule option sans en avoir fait l'inventaire ? Quelle a été la qualité de la décision finale ?*

Cinquième principe — Le sponsor s'engage sur la Phase A, pas sur la Phase F

Le dernier piège classique consiste à obtenir l'engagement du sponsor sur la totalité du cycle — un engagement de 18 mois, 12 millions d, 50 personnes-mois. Cet engagement est impossible à honorer et reste lettre morte.

La Phase A dure en pratique 6 à 8 semaines. Son livrable est léger (15 à 30 pages). Son coût est modéré (1 à 3 personnes-mois). **C'est ce sur quoi le sponsor doit s'engager**. Pas sur les 12 millions à venir, pas sur les 50 personnes-mois. Sur les 6 semaines et les 1-3 personnes-mois de la Phase A.

Une fois la Phase A livrée et validée, le sponsor dispose d'un cadrage clair pour engager — ou non — la Phase B. C'est ce qui rend la gouvernance soutenable : chaque phase est un engagement de faible ampleur, avec un point de validation explicite. Si le cadrage de la Phase A révèle que les ambitions initiales sont intenables, le sponsor peut redéfinir le périmètre avant que des millions ne soient engagés.

Chez Discount Voyages, Bertrand a obtenu un engagement ferme du COMEX sur 8 semaines de Phase A. Le périmètre a été réduit après le cadrage (la troisième option — outil back-office — s'est avérée être un prérequis aux deux autres, et a été engagée en premier). Sans ce cadrage progressif, les trois options auraient été engagées en parallèle, et la moitié du budget aurait été gaspillée.

***Et toi ?** Sur ton dernier cycle, quel engagement formel as-tu obtenu de ton sponsor ? Sur 8 semaines ou sur 18 mois ? L'engagement correspondait-il à ce que tu pouvais réellement livrer dans la période ?*

Application — Mon Business Scenario

À rédiger séance tenante. Six sections précises. À afficher à un endroit visible et à confier à deux personnes de confiance.

1. **Le problème** — Quel est le problème concret que mes parties prenantes essaient de résoudre ? En une phrase, formulé du point de vue de l'act, (pas du point de vue de la DSI).
2. **Les acteurs humains** — Qui sont les trois à cinq parties prenantes principales, et quel est leur intérêt personnel dans le problème ? Nommer les personnes réelles, pas les fonctions abstraites.
3. **Les acteurs informatiques** — Quels deux à quatre systèmes existants jouent un rôle dans le problème ? Décrire brièvement leur état actuel et leurs limites.
4. **Les intérêts** — Pour chaque acteur, quel est son critère de succès ? Comment le mesure-t-on aujourd'hui ?
5. **Les dépendances** — Quel événement déclenche le problème ? Quelle séquence d'événements le résoudrait ?
6. **La validation** — Avec qui vais-je tester ce scénario avant de l'engager dans la Phase A ?

Ce document d'une page est votre premier livrable de Business Scenario. C'est le matériau brut de votre Phase A.

Encadré Au lecteur / À l'animateur

Question d'ouverture : « Sur ton prochain projet d'architecture, combien d'exigences sont formalisées sous forme SMART aujourd'hui ? »

Profils à observer :

- **Le récitant qui connaît Phase A mais ne l'a jamais pratiquée.** Cite TOGAF, énumère les livrables, n'a jamais tenu un Business Scenario. *Question* : « Quel est ton dernier Statement of Architecture Work ? »
- **Le politicien qui cherche l'option qui plaît plutôt que l'option vraie.** Adapte son cadrage aux desiderata du COMEX. *Question* : « Sur quel indicateur ton cadrage repose-t-il en dernier ressort ? »
- **Le projeteur qui veut passer directement à la cible.** Trouve la Phase A trop abstraite, veut dessiner. *Question* : « Quel est le risque si tu n'alignes pas les parties prenantes avant de proposer une cible ? »
- **L'isolement du sponsor.** N'a personne pour valider sa Phase A, signe seul. *Question* : « Avec qui partages-tu ton Statement of Architecture Work avant de le présenter ? »

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les détails de la Phase B. Rappeler que la Phase A est un acte de **cadrage politique**, pas un exercice de modélisation.
- Ne pas confondre Phase A et étude d'opportunité. L'étude d'opportunité peut être antérieure à la Phase A. La Phase A commence une fois l'opportunité validée.

Cas pratique — Construction d'un Schéma Directeur Omnicanal (Retail)

Contexte sectoriel : chaîne de distribution retail, 3 000 employés, 250 magasins physiques, présence e-commerce en croissance. L'objectif stratégique est de **passer de 15 % à 30 % du chiffre d'affaires réalisé en ligne** dans les trois prochaines années. Le marché est dominé par des pure-players digitaux (Amazon, Cdiscount, Veepee) qui ont élevé les standards de réactivité client.

Diagnostic de l'existant. L'architecture en silos provoque un **désalignement critique** : **4 % des commandes web sont annulées** car le produit, affiché disponible, a été vendu en magasin physique entre-temps. Le coût d'opportunité et l'atteinte à l'image de marque imposent une unification immédiate. Le système de stock actuel fonctionne par magasin : chaque point de vente gère son propre inventaire, avec un délai de consolidation de 24 heures vers le siège.

Parties prenantes clés :

- **Directeur Retail** (pouvoir élevé, intérêt élevé) — co-décideur de la stratégie omnicanal.
- **Responsable E-commerce** (pouvoir élevé, intérêt élevé) — moteur du projet.
- **RSSI** (pouvoir élevé, intérêt moyen) — devient partie prenante pivot : l'ouverture du SI de stock via API expose l'entreprise à de nouveaux vecteurs d'attaque (API tierces, comparateurs, partenaires web).
- **Directeurs de magasins** (pouvoir moyen, intérêt élevé) — craignent la perte de visibilité sur leur stock et la complexité des nouveaux processus.
- **Équipes logistiques** (pouvoir faible, intérêt élevé) — bénéficiaires du projet mais sous-informés.

Méthodologie de construction du Schéma Directeur.

Le passage du AS-IS au TO-BE repose sur les « commandements de l'architecte » :

1. **L'exposition par API est le seul vecteur d'accès aux données de stock.** Aucun accès direct (« SQL direct ») aux bases de données ne sera toléré pour garantir l'intégrité des flux.
2. **Single Source of Truth** : le stock central est le maître. Aucun système local ne peut surcharger une disponibilité sans synchronisation immédiate.
3. **L'indépendance des plateformes** : la solution doit permettre d'intégrer de nouveaux transporteurs en moins de quatre semaines, sans modifier le cœur du système.

Construction étape par étape :

1. **Analyse de l'existant (As-Is)** : cartographier les sources de données stock, les flux de mise à jour, les fréquences, les silos (par magasin, par enseigne, par transporteur).
2. **Définition des objectifs cibles** : « Zéro rupture web sur 95 % du catalogue d'ici 18 mois », « Réduction du délai de mise à jour stock de 24h à 5 minutes ».
3. **Établissement des principes d'architecture** : API-first, event-driven pour les événements de stock, observabilité unifiée.
4. **Description de la cible (To-Be)** : plateforme de stock unifiée avec Event Sourcing, API Gateway, propagation temps réel vers tous les canaux (web, mobile, magasin, comparateurs).
5. **Trajectoire** : trois vagues (quick wins / fondations / déploiement massif).
6. **Gouvernance** : arbitrage des écarts via Architecture Review Board intégrant le RSSI dès la première revue.

Livrables attendus de l'apprenant (Schéma Directeur de trois pages) :

- **Page 1** : Contexte stratégique, parties prenantes, principes d'architecture.
- **Page 2** : Architecture cible avec schéma de flux, API exposées, événements émis.
- **Page 3** : Roadmap sur 18 mois, avec critères de succès chiffrés.

Pièges classiques à éviter :

- Tomber dans la sur-modélisation de l'existant. Ne pas viser l'exhaustivité ; viser la **valeur métier de la vision cible**.
- Minimiser l'enjeu de cybersécurité. Une API exposée sans protection devient le point d'entrée des attaquants.
- Négliger la communication vers les directeurs de magasin, qui peuvent bloquer le projet par inertie.

Critères d'évaluation : le SD est validé si l'apprenant (1) propose une cible simplifiée plutôt qu'exhaustive, (2) chiffre le gain attendu (réduction du taux d'annulation web), (3) identifie un chemin critique d'intégration en moins de quatre semaines pour un nouveau transporteur.

Ce que la leçon 5 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- **Un Business Scenario** formalisé selon les 6 étapes G176e, en une page.
- **Un Diagramme de Concept de Solution** avec trois options au maximum, présentées de manière comparable (coût, délai, risque, bénéfice).
- **Un sponsor identifié** qui pourra valider votre Phase A dans les 8 semaines.

Sources de cette leçon : TOGAF Series Guide to Business Scenarios (G176e) · Desfray & Raymond, *TOGAF en pratique* (chap. 6) · Cas Discount Voyages (anonymisé)

Leçon suivante : **L5 — L'Architecture Métier (Phase B) : la Business Capability Map.**

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Quel est le rôle central de la Phase A selon l'Open Group ?

A2 Qu'est-ce que l'« architecture après décision » ? Pourquoi la Phase A est-elle son antidote ?

A3 Qu'est-ce qu'un Business Scenario selon le guide TOGAF G176e ? Quelles sont ses 6 étapes ?

A4 Que signifie SMART pour un objectif d'architecture ? Pourquoi la formulation SMART cosmétique est-elle un piège ?

A5 Qu'est-ce que le Diagramme de Concept de Solution ? Quelle est sa fonction dans la Phase A ?

A6 Pourquoi l'engagement du sponsor doit-il porter sur la Phase A seulement, et non sur l'ensemble du cycle ADM ?

B — AUTO-EVALUATION

B1 Sur ton prochain projet d'architecture, combien d'exigences sont formalisées sous forme SMART aujourd'hui ? Lesquelles sont vraiment mesurables ?

B2 Combien de fois as-tu été appelé pour « documenter » une décision déjà prise ? Combien de fois as-tu accepté ce rôle ?

B3 Sur ton dernier projet, as-tu formalisé les exigences en Business Scenarios structurés avant de produire le moindre schéma ?

B4 Sur ton dernier livrable d'architecture, combien d'objectifs sont SMART au sens propre ? Lesquels ne le sont qu'en apparence ?

B5 Sur ton dernier projet, as-tu produit un schéma haut-niveau présentant plusieurs options en lice, ou as-tu foncé sur une seule option ?

B6 Sur ton dernier cycle, quel engagement formel as-tu obtenu de ton sponsor ? Sur 8 semaines ou sur 18 mois ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Rédiger un Business Scenario formalisé selon les 6 étapes G176e, en 1 page. (Échéance : J+7.)

C2 Produire un Diagramme de Concept de Solution avec 3 options maximum, présentées de manière comparable. (Échéance : J+14.)

C3 Identifier un sponsor capable de valider ma Phase A dans les 8 semaines. (Échéance : J+7.)

C4 Conduire 3 entretiens structurés (Business Scenarios) avec mes parties prenantes principales. (Échéance : J+21.)

C5 Tester chaque objectif « SMART » en demandant : « comment mesure-t-on cet objectif aujourd'hui, et qui a accès à cette mesure ? » (Échéance : J+10.)

C6 Obtenir l'engagement formel du sponsor sur la Phase A (8 semaines, 1-3 personnes-mois). (Échéance : J+30.)

L'Architecture Métier (Phase B) : la Business Capability Map

Architecture d'Entreprise

II — LA PROFONDEUR

« *Capability-Based Planning shifts the focus from projects to capabilities — what the enterprise must do, not what it builds.* » — The Open Group, TOGAF Series Guide — Business Capability Planning (G233e)

« *Décrire ce que l'organisation fait, avant de décrire comment elle le fait.* »

Vidéo d'introduction

 Télécharger cette leçon en PDF

Le platane qui cache la forêt

Sophie arrive en septembre 2019 comme architecte d'entreprise senior chez GreenTech, une ETI de 1 200 salariés dans le secteur de l'énergie renouvelable. La DSI a dix-huit mois d'historique de schémas d'architecture en tout genre : trois applications de cartographie, 47 documents ArchiMate non datés, une « cartographie SI » PowerPoint qui n'a jamais été mise à jour depuis sa création. Sophie passe ses trois premières semaines à interviewer les acteurs, à collecter les documents existants, à reconstituer l'histoire.

Le huitième jour, elle découvre le pot aux roses. Lors d'une réunion avec la directrice des opérations, celle-ci lui montre un schéma qu'elle a dessiné elle-même sur un coin de tableau blanc : « *Voilà comment fonctionne GreenTech.* » Le schéma fait neuf boîtes. Neuf capacités métier. Elles se lisent comme suit :

1. **Prospecter** de nouveaux clients
2. **Contractualiser** avec les clients
3. **Concevoir** les installations
4. **Construire** les installations
5. **Exploiter** les installations
6. **Maintenir** les installations
7. **Facturer** les clients
8. **Recouvrer** les paiements
9. **Innover** sur les nouvelles technologies

Sophie a passé huit jours à reconstituer un puzzle d'objets techniques, de serveurs, d'applications, de processus. Il lui a suffi d'un quart d'heure avec une opérationnelle pour obtenir **la véritable carte du territoire** — la carte des capacités que l'organisation doit maîtriser pour exister. Neuf capacités. Pas dix-sept applications, pas trois applications de cartographie, pas quarante-sept documents. Neuf capacités.

Le choc est salutaire. Sophie comprend ce que l'Open Group appelle **Capability-Based Planning** (TOGAF Series Guide

G233e) : la Phase B ne consiste pas à modéliser les processus, ni à dessiner des applications, ni à lister les serveurs. Elle consiste à **identifier les capacités que l'organisation doit maîtriser pour remplir sa mission**, puis à les organiser, les hiérarchiser, et les relier.

La Business Capability Map est l'outil qui rend cette pratique concrète. C'est une carte — un schéma — qui représente toutes les capacités métier de l'organisation, regroupées par domaines (marketing, vente, opérations, finance, RH, etc.). C'est le premier livrable canonique de la Phase B, et c'est ce que cette leçon cherche à vous apprendre à produire.

Voici à quoi ressemble une Capability Map structurée à trois niveaux (Domaine → Macro-capacité → Capacité opérationnelle) :

Premier principe — Capability ≠ Processus ≠ Application

L'une des confusions les plus coûteuses de la Phase B consiste à confondre trois objets très différents : les capacités, les processus, et les applications. Cette confusion produit des cartographies qui ne sont ni des capacités, ni des processus, ni des applications — des objets hybrides qui ne servent personne.

L'Open Group distingue clairement les trois dans son guide de référence (G233e). Une **capacité** est ce que l'organisation *doit savoir faire*. Elle est stable dans le temps — une mutuelle « sait facturer » depuis sa création, et continuera dans vingt ans. Elle ne se réduit pas à un processus unique (on peut facturer de plusieurs manières) ni à une application unique (on peut facturer avec plusieurs outils).

Un **processus** est *comment* l'organisation exerce ses capacités. Il décrit une séquence d'activités coordonnées par des acteurs et soutenues par des applications. Plusieurs processus peuvent contribuer à la même capacité. La capacité « facturation » peut s'exercer via un processus de facturation B2B (pour les grands clients), un processus de facturation B2C (pour les particuliers), un processus de facturation de prestations ponctuelles. Tous servent la même capacité.

Une **application** est ce que l'informatique *fournit* comme support aux processus. Une même application peut servir plusieurs processus, et donc plusieurs capacités. Un ERP peut servir la facturation, la comptabilité, la gestion des stocks — trois capacités distinctes, soutenues par un même système.

La contrefaçon la plus courante consiste à dessiner un schéma où les boîtes sont étiquetées « capacité » alors qu'elles décrivent des processus ou des applications. Le schéma « Gestion des contrats » ou « Progiciel SAP » est une capacité-écran : derrière ces mots se cachent soit des processus, soit des applications. La parade est de **tester chaque boîte** par la question : « cette boîte reste-t-elle valide dans cinq ans ? ». Si oui, c'est une capacité. Si non, c'est un processus ou une application.

***Et toi ?** Sur ta dernière « cartographie SI », combien de boîtes sont des capacités, combien des processus, combien des applications ? La distinction est-elle claire dans ton organisation ?*

Deuxième principe — La hiérarchie des capacités évite le plat unique

Une Business Capability Map qui ne hiérarchise pas ses capacités produit un « plat unique » — une longue liste plate où tout est au même niveau. Cette structure perd sa valeur stratégique dès qu'on dépasse la trentaine de capacités.

L'Open Group recommande une hiérarchie à **trois niveaux** :

- **Niveau 0 — Domaine** — les grandes familles (Marketing, Ventes, Opérations, Finance, RH, IT, Stratégie...)
- **Niveau 1 — Macro-capacité** — les grandes fonctions au sein d'un domaine (Marketing → Prospection, Communication, Fidélisation)

- **Niveau 2 — Capacité** — l'unité opérationnelle minimale (Marketing → Prospection → Génération de leads, Qualification, Conversion)

Cette hiérarchie permet plusieurs usages :

- **Vue macro** (Niveau 0-1) — pour les discussions COMEX, les revues annuelles, les arbitrages budgétaires
- **Vue opérationnelle** (Niveau 1-2) — pour les décisions d'investissement, les analyses de cible, la cartographie applicative
- **Vue de drill-down** — pour analyser l'écart entre As-Is et To-Be sur une capacité précise

La contrefaçon à éviter est le plat unique. Une organisation qui a 80 capacités au même niveau produit une carte illisible, qui ne sert qu'à montrer qu'on a 80 boîtes — pas à prendre des décisions. La parade est de **résister à la tentation du détail** : tant qu'une capacité n'est pas l'objet d'une décision ou d'une discussion active, elle reste au niveau supérieur.

***Et toi ?** Combien de niveaux ta dernière cartographie a-t-elle ? Combien de capacités sont au niveau le plus fin ? Ce chiffre est-il cohérent avec le nombre de décisions prises sur la base de cette cartographie ?*

Troisième principe — La Capability Map n'est pas un projet, c'est une pratique

L'un des travers les plus coûteux consiste à traiter la Business Capability Map comme un livrable de projet — un document qu'on produit en trois mois et qu'on n'actualise plus jamais. Cette approche produit des cartographies qui pourrissent sur étagères.

L'Open Group recommande que la Capability Map soit **vivante** :

- **Versionnée** — un historique des versions, daté, archivé
- **Maintenue** — une équipe (même petite) responsable des mises à jour
- **Consultée** — utilisée dans les revues de gouvernance, les arbitrages budgétaires, les décisions d'investissement
- **Évolutive** — révisée annuellement, avec un cycle de validation léger

Sans cette discipline, la cartographie devient un monument commémoratif d'une intention passée. Avec cette discipline, elle devient un outil de gouvernance — un référentiel commun qui structure les décisions.

Chez GreenTech, Sophie a institué une revue trimestrielle de la Capability Map — trois réunions de deux heures par an, avec les propriétaires de domaine (Directeurs Marketing, Ventes, Opérations, etc.). Chaque révision ajoute une à trois capacités, en supprime une à deux obsolètes, et raffine le libellé d'une demi-douzaine. La cartographie reste à jour sans peser sur l'équipe d'architecture.

***Et toi ?** À quand remonte la dernière mise à jour de ta cartographie ? Combien de fois a-t-elle été consultée pour une vraie décision cette année ? Si la réponse est « moins de trois fois », ta cartographie est probablement un monument commémoratif.*

Quatrième principe — Le Capability Planning oriente l'investissement, pas l'inverse

Le piège classique consiste à déduire la cartographie des applications existantes, plutôt que d'orienter les applications

par la cartographie des capacités. Cette inversion transforme la Phase B en documentation du SI, pas en architecture.

Le **Capability-Based Planning** (G233e) prescrit l'ordre suivant :

1. **Identifier les capacités** — ce que l'organisation doit savoir faire
2. **Évaluer la maturité de chaque capacité** — où en sommes-nous (1 à 5) sur chacune
3. **Identifier les écarts** — où la maturité est insuffisante par rapport aux objectifs
4. **Hiérarchiser les investissements** — par capacité, pas par projet applicatif

L'inverse — déduire les capacités des applications — produit des capacités « il y a SAP pour la finance », « il y a Salesforce pour le CRM ». Ces formulations masquent un problème fondamental : on ne sait pas ce que l'organisation doit faire, on sait seulement ce qu'elle fait avec ses outils actuels. Le Capability Planning sert précisément à échapper à cette myopie.

L'Open Group l'explique ainsi dans le guide G233e : « *Capability-Based Planning shifts the focus from projects to capabilities — what the enterprise must do, not what it builds* ». Le glissement de « projects » vers « capabilities » est l'inversion stratégique qui distingue une AE moderne d'une simple documentation du SI existant.

La contrefaçon à éviter : se laisser tenter par un Capability Planning dont les capacités sont si vagues qu'elles n'orientent aucun investissement (« Innovation », « Excellence opérationnelle »). Ces capacités-écrans ne servent qu'à orner des slides. La parade est de tester chaque capacité par la question « quel investissement cette capacité oriente-t-elle ? ». Si la réponse est floue, la capacité est trop abstraite — il faut la décomposer.

***Et toi ?** Sur ton dernier plan d'investissement SI, les projets étaient-ils classés par capacité métier, ou par application ? Si par application, quelle visibilité avais-tu sur la couverture réelle des capacités ?*

Cinquième principe — Les Value Streams complètent les Capabilities

La Business Capability Map montre ce que l'organisation doit savoir faire. Les Value Streams montrent comment elle crée de la valeur, de bout en bout, à travers ces capacités. Les deux sont complémentaires.

L'Open Group publie un guide dédié aux Value Streams (G178e). Un Value Stream est une **séquence de bout en bout** qui traverse plusieurs capacités pour créer de la valeur pour un client (interne ou externe). Pour GreenTech, le Value Stream « Acquisition d'un nouveau client B2B » traverse les capacités :

1. **Prospecter** (génération de leads)
2. **Qualifier** (scoring)
3. **Contractualiser** (négociation)
4. **Concevoir** (étude technique)
5. **Construire** (installation)
6. **Facturer** (première facture)
7. **Recouvrer** (paiement)

Sept capacités traversées. Le Value Stream rend visible le **parcours complet** — ce que la Capability Map seule ne montre pas. Une capacité peut être mature (note 5/5) sans que le Value Stream qui la traverse soit performant — par exemple si les transitions entre capacités sont maladroites, mal outillées, mal gouvernées.

L'articulation entre Capability Map et Value Streams est au cœur de la Phase B :

- **La Capability Map** sert à structurer le **portefeuille d'investissement** : où investir pour combler les écarts de maturité

- **Les Value Streams** servent à structurer les **chantiers opérationnels** : où fluidifier les transitions pour améliorer l'expérience client

La contrefaçon à éviter : se contenter de la Capability Map sans Value Streams. On investit alors pour « améliorer la capacité de facturation » sans réaliser que la vraie douleur du client est dans la transition entre « facturation » et « recouvrement ». La Capability Map seule est myope sur les frictions.

Et toi ? Combien de Value Streams as-tu identifiés pour ton organisation ? Traversent-ils les capacités critiques de bout en bout ? As-tu déjà mesuré le temps de cycle d'un Value Stream complet ?

Application — Ma Capability Map

À rédiger séance tenante. Trois livrables précis, à afficher et à confier à deux personnes de confiance.

1. **Niveau 0 — Domaines** — Les six à douze grandes familles de capacités de ton organisation. Marketing, Ventes, Opérations, Finance, RH, IT, Stratégie... Adapte au contexte.
2. **Niveau 1 — Macro-capacités** — Pour chaque domaine, les quatre à huit grandes fonctions. Marketing → Prospection, Communication, Fidélisation, Études, Pricing. Etc.
3. **Niveau 2 — Capacités opérationnelles** — Pour chaque macro-capacité, les trois à cinq unités opérationnelles minimales. Marketing → Prospection → Génération de leads, Qualification, Conversion, Suivi.

|> Cette carte est ton premier livrable de Phase B. C'est elle qui orientera tes investissements, tes analyses d'écart, et tes chantiers opérationnels.

Module complémentaire — La friction AE × Agilité au niveau métier

La Phase B produit une **cartographie stable** des capacités métier (L0 → L1 → L2). Les méthodes agiles produisent un **flux d'événements** changeants (Epics, Features, Stories). Comment ces deux visions s'articulent-elles sans se neutraliser ?

Le Capability Map n'est pas un backlog

Une erreur fréquente consiste à confondre la **Capability Map** (cartographie des capacités stables) avec un **Product Backlog** (liste de choses à faire changeante). Ces deux artefacts ont des objectifs différents :

- La Capability Map décrit **ce que l'organisation sait faire** (ou devrait savoir faire) à un horizon de 3 à 5 ans. Elle est révisée annuellement, pas quotidiennement.
- Le Product Backlog décrit **ce que l'organisation va faire** dans les prochaines semaines. Il change à chaque sprint.

Le Capability Map est au Product Backlog ce que la **stratégie** est au **plan d'action tactique** : la première fixe le cap, le second adapte les moyens au terrain.

Le flux de valeur métier (Value Stream Mapping)

Pour relier la Capability Map aux trains agiles, l'AE utilise le concept de **flux de valeur métier** (Development Value Stream, concept SAFe aligné TOGAF). Un flux de valeur est la **séquence des capacités nécessaires** pour délivrer un résultat métier donné.

Exemple Mutuelle Plus : le flux de valeur « Remboursement santé » traverse les capacités suivantes : 1. Réception de la demande (canal web, agence, courrier). 2. Vérification des droits (capacité « Adhésion & Éligibilité »). 3. Contrôle de cohérence (capacité « Détection de fraude »). 4. Calcul du remboursement (capacité « Gestion des prestations »). 5. Mise en paiement (capacité « Comptabilité »). 6. Notification à l'adhérent (capacité « Communication client »).

Ce flux **traverse** plusieurs domaines et donc plusieurs équipes agiles potentielles. C'est à l'AE de garantir que les **NFR** (performance, sécurité, conformité RGPD) sont respectées à chaque maillon du flux, et que les **dépendances inter-équipes** sont identifiées.

Le rôle de l'AE dans le PI Planning métier

Dans un PI Planning qui aligne plusieurs trains agiles sur un flux de valeur métier, l'AE intervient pour :

1. **Vérifier que les Features planifiées s'inscrivent dans la Capability Map.** Une Feature sans capacité cible est une Feature qui crée du code orphelin.
2. **Identifier les capacités sous-tension.** Une capacité mobilisée par 12 Features dans le même sprint est probablement un goulot d'étranglement à court terme. L'AE remonte l'alerte.
3. **Tracer les dépendances inter-domaines.** Si une Feature du domaine « Santé Individuelle » dépend d'une capacité du domaine « Réclamations », cette dépendance doit être visible, documentée, et priorisée dans le backlog.
4. **Aligner la dette technique avec la cible.** Si la Capability Map prévoit une cible « Détection de fraude automatisée » dans 18 mois, l'AE s'assure que les Enablers actuels construisent progressivement cette capacité, plutôt que de colmater les failles du présent.

Trois erreurs de friction AE × Agilité à éviter

1. **L'AE qui veut imposer une Capability Map rigide.** Le métier évolue, les capacités évoluent. L'AE qui refuse de réviser la cartographie tous les 12 mois devient un frein plutôt qu'un cadre.
2. **L'équipe agile qui ignore la Capability Map.** Une équipe qui développe sans regarder la cartographie crée du code qui ne s'inscrit dans aucune vision long terme. Six mois plus tard, personne ne sait pourquoi cette application existe.
3. **La confusion entre Feature et Capacité.** Une Feature (« Bouton de paiement en un clic ») n'est pas une Capacité (« Gestion des paiements »). L'AE doit maintenir cette distinction : la Capacité est stable, la Feature est ponctuelle.

Synthèse

L'AE ne s'oppose pas à l'agilité. Il l'ancre dans une **vision métier stable**. La Capability Map est le **contrat de cohérence** entre le long terme (stratégie, capacités) et le court terme (Features, sprints). Sans elle, l'agilité produit de la vitesse sans direction ; sans l'agilité, la cartographie reste théorique. Les deux se complètent.

Encadré Au lecteur / À l'animateur

Question d'ouverture : « Sur ton dernier projet d'architecture, as-tu modélisé des capacités, des processus, ou des applications ? La distinction est-elle claire ? »

Profils à observer :

- **Le cartographe du SI existant.** Décrit ce que les outils font, pas ce que l'organisation doit faire. *Question* : « Cette capacité reste-t-elle valide dans cinq ans ? »
- **Le stratège qui pense en catégories abstraites.** Produit des capacités « Excellence opérationnelle » qui n'orientent rien. *Question* : « Quel investissement concret cette capacité oriente-t-elle ? »
- **Le processeur qui modélise des processus sous le nom de capacités.** Confond les niveaux d'abstraction. *Question* : « Cette boîte est-elle un quoi ou un comment ? »

- **Le rêveur qui veut 200 capacités.** Vise l'exhaustivité, produit l'illisibilité. *Question : « Combien de décisions sont prises sur cette capacité dans l'année ? »*

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les outils (SAP, Salesforce, Office 365). Rappeler que la Capability Map est **indépendante des outils**.
- Ne pas confondre Capability Map et cartographie SI. La première est stable dans le temps ; la seconde est mouvante.

Ce que la leçon 6 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- **Votre Capability Map** à trois niveaux, validée par les propriétaires de domaine.
- **Trois Value Streams** critiques, identifiés de bout en bout.
- **Une évaluation de maturité** (1 à 5) pour chacune des macro-capacités.

Sources de cette leçon : TOGAF Series Guide to Business Capability Planning (G233e) · TOGAF Series Guide to Value Streams (G178e) · Desfray & Raymond, *TOGAF en pratique* (chap. 7) · Cas GreenTech (anonymisé)

Leçon suivante : **L6 — La Modélisation des Processus (BPMN) : la grammaire du comment.**

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Quelle est la différence entre une capacité, un processus, et une application ? Pourquoi cette distinction est-elle cruciale en Phase B ?

A2 Quels sont les trois niveaux hiérarchiques d'une Business Capability Map selon l'Open Group ? Quel usage correspond à chaque niveau ?

A3 Pourquoi une Capability Map doit-elle être vivante (versionnée, maintenue, consultée, évolutive) plutôt qu'un livrable de projet ?

A4 Qu'est-ce que le Capability-Based Planning ? En quoi diffère-t-il d'une approche par projet applicatif ?

A5 Qu'est-ce qu'un Value Stream (TOGAF G178e) ? Comment complète-t-il la Capability Map ?

A6 Quel est le test qui distingue une capacité d'un processus ou d'une application ?

B — AUTO-EVALUATION

B1 Sur ta dernière « cartographie SI », combien de boîtes sont des capacités, combien des processus, combien des applications ? La distinction est-elle claire ?

B2 Combien de niveaux ta dernière cartographie a-t-elle ? Combien de capacités au niveau le plus fin ? Ce chiffre est-il cohérent avec le nombre de décisions prises ?

B3 À quand remonte la dernière mise à jour de ta cartographie ? Combien de fois a-t-elle été consultée pour une vraie décision cette année ?

B4 Sur ton dernier plan d'investissement SI, les projets étaient-ils classés par capacité métier ou par application ? Quelle visibilité avais-tu sur la couverture réelle des capacités ?

B5 Combien de Value Streams as-tu identifiés pour ton organisation ? Traversent-ils les capacités critiques de bout en bout ? As-tu déjà mesuré leur temps de cycle ?

B6 Sur ton dernier projet d'architecture, as-tu modélisé des capacités, des processus, ou des applications ? La distinction est-elle claire dans ton organisation ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Rédiger ma Capability Map à 3 niveaux : 6-12 domaines, 4-8 macro-capacités par domaine, 3-5 capacités opérationnelles par macro. (Échéance : J+14.)

C2 Valider la Capability Map avec les propriétaires de domaine (1 réunion par domaine, 1-2h chacune). (Échéance : J+30.)

C3 Identifier 3 Value Streams critiques traversant la Capability Map de bout en bout. (Échéance : J+21.)

C4 Évaluer la maturité (1 à 5) de chaque macro-capacité, et identifier les 3 écarts les plus significatifs. (Échéance : J+30.)

C5 Tester chaque capacité par la question « quel investissement concret cette capacité oriente-t-elle ? » — supprimer les capacités-écrans qui n'orientent rien. (Échéance : J+10.)

C6 Mesurer le temps de cycle d'un Value Stream complet (du déclencheur à la valeur livrée au client). (Échéance : J+45.)

Les Processus (BPMN) : l'action opérationnelle

Architecture d'Entreprise

II — LA PROFONDEUR

« Processes are the invisible engines of the organization. » — Michael Hammer & James Champy, *Reengineering the Corporation* (1993)

« Un bon processus n'est pas celui qui est bien dessiné. C'est celui qui transforme la décision en exécution. »

Vidéo d'introduction

📄 Télécharger cette leçon en PDF

Le tunnel d'achat que personne ne comprenait

En 2018, la DSI de Discount Voyages avait un problème de processus qu'elle n'arrivait pas à résoudre. Le tunnel d'achat en ligne — du clic sur « Réserver » à la confirmation de réservation — était constitué de douze étapes enchaînées par six systèmes différents. Chaque étape faisait l'objet d'une documentation PowerPoint locale, d'un wiki Confluence dépassé, et parfois d'un schéma Visio qui n'avait plus rien à voir avec la réalité. Les incidents de production se multipliaient. Le service client recevait des appels de clients bloqués à des étapes différentes. Personne ne savait expliquer pourquoi.

Camille, l'architecte d'entreprise, est appelée. Elle commence par un geste inhabituel : ne pas regarder la documentation, mais **interviewer les opérateurs**. Elle passe une semaine avec l'équipe du service client, deux jours avec les développeurs du site web, une demi-journée avec la comptabilité. Elle reconstitue le tunnel d'achat tel qu'il est réellement vécu — pas tel qu'il est documenté. Elle découvre quatre choses :

1. **Le tunnel documenté n'existe pas.** Les étapes documentées dans les PowerPoint ne correspondent plus au comportement réel des systèmes depuis la dernière mise à jour du site.
2. **Les étapes implicites sont les plus longues.** Trois étapes cruciales ne sont nulle part documentées — la validation du panier par l'anti-fraude, la mise en attente du paiement 3D Secure, la double-saisie comptable post-réservation.
3. **Le schéma qu'on dessine est un acte politique.** Chaque service veut que son étape apparaisse comme étant la plus rapide, la plus fiable, la plus moderne. Dessiner le vrai processus, c'est nommer les responsabilités qui fâchent.
4. **Le BPMN est le bon outil, mais pas pour la raison qu'on croit.** Le BPMN n'est pas fait pour « documenter ». Il est fait pour **rendre le flux discutable** — pas seulement visible.

Camille passe six semaines à produire un BPMN du tunnel d'achat, en étroite collaboration avec les six services concernés. Le schéma final fait sept pages. Il remplace les douze PowerPoint locaux, les trois wikis Confluence, les cinq Visio obsolètes. Mieux : il devient la référence pour les comités de gouvernance, les revues d'incident, les nouveaux projets d'évolution.

C'est ce que cette leçon cherche à vous transmettre : **comment utiliser BPMN pour rendre les flux discutables, et**

pas seulement visibles.

Le BPMN qui suit modélise le tunnel d'achat de Discount Voyages tel que Camille l'a reconstitué — 5 activités, 4泳lane, 1 passerelle de décision. Cette vue macro est volontairement épurée pour rester discutable :

Premier principe — BPMN ≠ UML ≠ organigramme : chaque outil a sa fonction

L'un des pièges les plus coûteux consiste à utiliser un mauvais outil pour modéliser les processus. Cette erreur produit des documents hybrides qui ne servent personne, et qui sont abandonnés au bout de six mois.

Le choix de l'outil de modélisation de processus n'est pas anodin. Quatre grandes familles coexistent, avec des fonctions différentes :

- **L'organigramme** — représente une structure hiérarchique de pouvoir. Utile pour les organigrammes RH, pas pour les processus.
- **UML Activity Diagram** — représente un comportement algorithmique. Précis, formel, adapté aux développeurs. Rébarbatif pour les opérationnels.
- **BPMN 2.0 (OMG)** — représente un flux métier avec une grammaire normalisée. Précis pour les développeurs, lisible pour les opérationnels. Conçu pour l'exécution (BPMN est aussi un format exécutable par les moteurs de workflow).
- **Cartographie des processus SIPOC** (Supplier-Input-Process-Output-Customer) — représente les interfaces d'un processus. Vue macroscopique, pas de flux détaillé.

La contrefaçon classique consiste à dessiner un organigramme en prétendant que c'est un processus, ou à dessiner un diagramme d'activité UML en prétendant que c'est un BPMN. La parade est de tester le document final par deux questions :

- « *Un opérationnel non-technique peut-il lire ce document et pointer du doigt son étape ?* » — si non, c'est trop formel pour un processus.
- « *Un développeur peut-il extraire de ce document les conditions de chaque transition ?* » — si non, c'est trop vague pour un processus.

BPMN est l'outil qui répond aux deux questions par l'affirmative. C'est pour cela qu'il s'est imposé comme standard international depuis 2005, maintenu par l'Object Management Group (OMG) et promu par l'OCEB (Object Management Group Certified Expert in BPM).

***Et toi ?** Sur ton dernier processus modélisé, as-tu utilisé un organigramme, un diagramme UML, un BPMN, ou autre chose ? Le document final était-il lisible par les opérationnels ET exploitable par les développeurs ?*

Deuxième principe — Macro-processus dynamique vs domaine fonctionnel stable

Le Capability Map (L5) montre ce que l'organisation doit savoir faire. Les processus montrent comment elle le fait. Ce sont deux objets différents, et il faut savoir les articuler.

L'Open Group distingue clairement les deux dans ses guides :

- **Le Capability Map** est **stable dans le temps** — une mutuelle « sait facturer » depuis 1948 et continuera dans

vingt ans. Le Capability Map est révisé annuellement.

- **Les processus sont dynamiques** — ils changent à chaque évolution des outils, des réglementations, des pratiques commerciales. Les processus sont révisés en continu.

Cette distinction a une implication directe : **il ne faut pas chercher à cartographier exhaustivement tous les processus**. Le Capability Map est exhaustif ; les processus sont échantillonnés. On ne modélise en BPMN que les processus qui font l'objet d'une décision, d'une réorganisation, ou d'une difficulté opérationnelle. Les autres processus restent dans la mémoire collective.

La contrefaçon classique consiste à produire un « référentiel des processus » exhaustif de 200 processus documentés en BPMN, qui n'est jamais utilisé. C'est l'équivalent, en processus, d'une cartographie applicative exhaustive qu'on ne consulte jamais. La parade est de **ne modéliser en BPMN que les processus en jeu** dans une décision, et de re-modéliser quand la décision évolue.

***Et toi ?** Combien de processus as-tu modélisés en BPMN sur ton dernier projet ? Ce nombre est-il cohérent avec le nombre de décisions qui ont été prises sur la base de cette modélisation ?*

Troisième principe — *Ruthless minimization* : ne modéliser que ce qui sert la décision

La tentation du modélisateur est de tout représenter. La règle d'or en BPMN est inverse : ne modéliser que ce qui sert la décision en cours.

Cette règle — qu'on peut appeler *ruthless minimization* — s'applique à trois niveaux :

- **Niveau activité** — ne pas détailler les sous-étapes d'une activité. Une activité « Vérifier le dossier client » est une boîte noire ; on ne dessine pas ses trois sous-étapes. Si le détail est utile à la décision, on le met dans une泳道 (swimlane) séparée.
- **Niveau泳道** — ne pas multiplier les泳道. Au-delà de cinq ou six participants, le BPMN devient illisible. Si le processus en compte plus, on découpe en sous-processus.
- **Niveau exception** — ne pas représenter toutes les exceptions. On représente l'exception qui est l'objet de la décision, pas celles qui sont déjà gérées par les outils (gestion d'erreur automatique, retry, etc.).

La contrefaçon classique consiste à dessiner un BPMN qui « montre tout » — toutes les étapes, tous les participants, toutes les exceptions. Le résultat est un schéma de trois mètres de large, illisible, qui décourage toute discussion. La parade est de demander avant chaque boîte ajoutée : « *cette information est-elle nécessaire à la décision en cours ?* » Si non, on retire la boîte.

Pour Discount Voyages, Camille a modélisé le tunnel d'achat avec **un seul BPMN principal** (sept pages, niveau activité), puis **trois BPMN de sous-processus** (un par sous-processus critique : paiement 3D Secure, validation anti-fraude, double-saisie comptable). Le tout est lisible. Le tout est discutable.

***Et toi ?** Sur ton dernier BPMN, combien d'activités as-tu représentées ? Combien de泳lane ? Combien d'exceptions ? Ces nombres sont-ils cohérents avec l'objectif de la décision qui a conduit à modéliser ?*

Quatrième principe — Top-down d'abord, bottom-up ciblé ensuite

L'ordre de modélisation compte. Modéliser d'abord les processus de bout en bout, puis zoomer sur les sous-processus critiques — pas l'inverse.

L'OCEB (Object Management Group Certified Expert in BPM) recommande un **ordre de modélisation en deux passes** :

- **Passe 1 — Top-down** — identifier les trois à cinq processus clés de l'organisation. Pour chacun, dessiner un BPMN de niveau macro (5 à 15 activités). Ce BPMN sert à la communication exécutive, à l'arbitrage des priorités, à la revue annuelle.
- **Passe 2 — Bottom-up ciblé** — pour les processus qui font l'objet d'une décision ou d'une réorganisation, zoomer. Dessiner un BPMN détaillé (20 à 50 activités), avec les泳lane, les exceptions, les indicateurs.

Cette double passe évite deux travers :

- **Le travers top-down-only** — un BPMN macro qui se contente de dire « il y a un processus de vente » sans jamais entrer dans le détail. Décision impossible.
- **Le travers bottom-up-only** — un BPMN détaillé d'un sous-processus sans vision d'ensemble. On ne sait pas si ce sous-processus est critique, négligeable, ou redondant.

Pour Discount Voyages, Camille a d'abord modélisé les cinq processus clés (réservation, paiement, facturation, service client, marketing) en BPMN macro (5-10 activités chacun). Puis elle a zoomé sur le tunnel d'achat en BPMN détaillé, parce que c'était l'objet de la décision. Les quatre autres processus clés restent au niveau macro.

***Et toi ?** Sur ton dernier projet de modélisation, as-tu démarré top-down puis zoomé, ou as-tu directement dessiné le détail d'un sous-processus ? Quel a été l'usage réel de chacun des BPMN produits ?*

Cinquième principe — Le BPMN comme contrat entre métier et IT

Le BPMN n'est pas un document technique, ni un document métier. C'est un contrat entre les deux — un acte commun qui engage les deux parties sur la même compréhension du flux.

La fonction politique du BPMN est aussi importante que sa fonction technique. Quand un BPMN est validé conjointement par un opérationnel et un développeur, les deux parties s'engagent sur :

- **Les activités** — qui fait quoi, dans quel ordre
- **Les transitions** — sous quelles conditions le flux passe d'une activité à l'autre
- **Les exceptions** — que se passe-t-il si l'activité échoue, si l'entrée n'arrive pas, si le délai est dépassé
- **Les indicateurs** — comment mesure-t-on la performance du processus

Cet engagement est **réversible** : on peut modifier le BPMN si les conditions changent, mais chaque modification est une décision documentée. C'est ce qui distingue un BPMN de un document Word oublié.

La contrefaçon classique consiste à produire un BPMN technique sans validation opérationnelle — un schéma que personne ne reconnaît comme étant le sien. Le BPMN reste sans usage, et finit par être abandonné. La parade est d'**associer systématiquement un opérationnel à chaque modélisation**, et de ne jamais figer un BPMN sans signature opérationnelle.

Pour Discount Voyages, Camille a tenu six ateliers d'une demi-journée chacun, un par service concerné. Chaque atelier a abouti à une version signée du BPMN. C'est cette signature qui a rendu le BPMN opérationnel — pas le schéma lui-même.

***Et toi ?** Sur ton dernier BPMN, qui l'a validé ? Un développeur seul, un opérationnel seul, ou les deux ensemble ? Combien de fois le BPMN a-t-il été consulté pour une vraie décision depuis sa validation ?*

Application — Mon BPMN

À produire séance tenante. Un seul processus réel de ton organisation, modélisé selon les cinq principes de cette leçon.

1. **Choisir le processus** — un processus qui fait l'objet d'une décision, d'une réorganisation, ou d'une difficulté opérationnelle.
2. **Dessiner le BPMN macro** — 5 à 15 activités, 3 à 5泳lane. Niveau d'abstraction permettant la discussion exécutive.
3. **Identifier les sous-processus critiques** — sur les 5 à 15 activités, lesquelles font l'objet d'une décision ou d'une difficulté ? (Une à trois activités maximum.)
4. **Dessiner le BPMN détaillé du sous-processus** — pour chaque sous-processus critique, 20 à 50 activités, avec exceptions et indicateurs.
5. **Valider avec un opérationnel** — un atelier d'une demi-journée, aboutissant à une signature commune.
6. **Diffuser le BPMN signé** — wiki d'architecture, comité de gouvernance, base de connaissance.

Ce BPMN est ton premier livrable de processus. Il sert de contrat entre ton service et les services partenaires.

Encadré Au lecteur / À l'animateur

Question d'ouverture : « Sur ton dernier processus critique, as-tu utilisé un organigramme, un diagramme UML, un BPMN, ou autre chose ? »

Profils à observer :

- **Le modélisateur exhaustif.** Veut tout représenter, produit des schémas illisibles. *Question :* « Cette activité est-elle nécessaire à la décision ? »
- **Le formaliste UML.** Produit des schémas précis que personne ne lit. *Question :* « Un opérationnel peut-il pointer du doigt son étape ? »
- **Le politique qui veut un schéma qui arrange.** Évite de représenter les responsabilités qui fâchent. *Question :* « Qui est responsable de cette activité sur ton BPMN ? »
- **L'isolé qui modélise seul.** N'associe aucun opérationnel. Produit des schémas sans usage. *Question :* « Qui a signé ton dernier BPMN ? »

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les outils (Visio, Bizagi, Camunda, etc.). Rappeler que l'outil est secondaire, le standard BPMN est central.
- Ne pas confondre BPMN et organigramme. Un processus décrit un flux, pas une hiérarchie.

Cas pratique — Optimisation du processus d'admission patient (Hôpital)

Contexte sectoriel : hôpital régional public, 1 200 lits, 280 000 admissions par an, 4 500 professionnels (médicaux,

soignants, administratifs). Le processus d'admission est critique : il conditionne la qualité de prise en charge, la facturation, la sécurité des soins (identitovigilance).

Diagnostic de l'existant. Le processus actuel souffre de **4,5 % d'erreurs d'identification**, avec un coût de traitement de **80 € par dossier erroné**. L'absence de contrats numérisés et les recherches manuelles permanentes paralysent l'accueil. Le délai médian entre l'arrivée du patient et la disponibilité du dossier administratif est de **45 minutes**, dont 30 minutes passées à rechercher des informations dans des classeurs physiques ou à reconstituer un identifiant à partir de documents incomplets.

Parties prenantes :

- **Chef de service des Admissions** (décisionnaire) — porte la responsabilité de la conformité et de la qualité.
- **Responsable Qualité** (décisionnaire) — valide les indicateurs et pilote les plans d'amélioration.
- **Secrétaires médicales et opérateurs d'admissions** (utilisateurs finaux critiques) — leur adhésion conditionne l'efficacité. L'ergonomie de l'interface métier est non négociable.
- **Service Médical** (consommateur du processus) — besoin d'un dossier patient fiable avant la consultation.
- **Régulateur Externe** (ARS, CNIL) — conformité RGPD et identitovigilance.

Démarche de modélisation BPMN 2.0.

L'atelier exige une modélisation rigoureuse, structurée par Pools et Lanes :

- **Pools (Piscines)** : distinguer le **Patient**, l'**Hôpital** et le **Régulateur Externe** (pour la conformité RGPD). Cette séparation rend visibles les frontières de responsabilité.
- **Lanes (Couloirs)** dans la Pool « Hôpital » : séparer l'**Accueil**, le **Service Médical** et la **Facturation**. Chaque lane représente un acteur organisationnel.

Flux TO-BE modélisé :

1. Le patient arrive à l'accueil, présente sa carte Vitale et une pièce d'identité.
2. Lecture automatique de la carte Vitale via un terminal NFC (suppression de la saisie manuelle).
3. Croisement avec le référentiel d'identitovigilance national (INS — Identifiant National de Santé) via les standards **HL7/FHIR**.
4. Si identité reconnue : dossier pré-rempli, validation en un clic. Si identité non reconnue : workflow de création avec vérification croisée.
5. Transmission du dossier au Service Médical en moins de 5 minutes (vs 30 minutes actuellement).
6. Génération automatique des données de facturation à partir du dossier validé.

Indicateurs cibles :

- Réduction du **temps d'attente** de 45 min à moins de 10 min.
- **Taux d'erreur d'identification** ramené de 4,5 % à moins de 1 %.
- **Coût par dossier erroné** divisé par 5.

Erreur type à éviter. Modéliser les écrans du logiciel au lieu du flux logique du patient. L'architecte modélise le **métier**, pas l'interface. Un BPMN qui décrit « cliquer sur le bouton X » est un organigramme applicatif déguisé.

Livrables attendus de l'apprenant :

1. Un BPMN 2.0 du processus d'admission avec trois pools et au moins trois lanes.
2. Une cartographie des risques avec parades (perte de pièce d'identité, patient inconscient, homonymie).
3. Un plan d'amélioration continue avec trois KPIs mesurables.

Critères d'évaluation : le BPMN est validé si l'apprenant (1) respecte les Pools/Lanes par acteur organisationnel, (2) introduit les standards HL7/FHIR, (3) propose des indicateurs chiffrés atteignables en 12 mois.

Ce que la leçon 7 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- **Un BPMN macro** d'un processus clé, validé par un opérationnel.
- **Un BPMN détaillé** d'un sous-processus critique, signé conjointement.
- **Une liste des 3 à 5 processus clés** de ton organisation, avec leur statut (modélisé / à modéliser / non applicable).

Sources de cette leçon : OMG, *Business Process Model and Notation (BPMN) Version 2.0* (2011) · OCEB 2, *Fundamental Level* · Hammer & Champy, *Reengineering the Corporation* (1993) · Cas Discount Voyages (anonymisé)

Leçon suivante : **L7 — L'Architecture des Données : la pyramide DAMA.**

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Quelle est la différence entre un organigramme, un diagramme d'activité UML, un BPMN, et un SIPOC ? Quel outil pour quel usage ?

A2 Pourquoi le Capability Map est-il stable dans le temps alors que les processus sont dynamiques ? Comment les articuler ?

A3 Qu'est-ce que la ruthless minimization en BPMN ? Comment s'applique-t-elle aux niveaux activité,泳lane, et exception ?

A4 Quelle est la différence entre une approche top-down (Inmon) et une approche bottom-up (Kimball) en BPMN ? Pourquoi l'approche hybride est-elle recommandée ?

A5 Pourquoi un BPMN doit-il être validé conjointement par un opérationnel et un développeur ? Quelle est sa fonction politique ?

A6 Quelle est la règle de limitation du nombre de泳lane ? Pourquoi cette règle existe-t-elle ?

B — AUTO-EVALUATION

B1 Sur ton dernier processus modélisé, as-tu utilisé un organigramme, un diagramme UML, un BPMN, ou autre chose ? Le document final était-il lisible par les opérationnels ET exploitable par les développeurs ?

B2 Combien de processus as-tu modélisés en BPMN sur ton dernier projet ? Ce nombre est-il cohérent avec le nombre de décisions prises ?

B3 Sur ton dernier BPMN, combien d'activités as-tu représentées ? Combien de泳lane ? Combien d'exceptions ? Ces nombres sont-ils cohérents avec l'objectif de la décision ?

B4 Sur ton dernier projet de modélisation, as-tu démarré top-down puis zoomé, ou as-tu directement dessiné le détail d'un sous-processus ? Quel a été l'usage réel de chaque BPMN ?

B5 Sur ton dernier BPMN, qui l'a validé ? Un développeur seul, un opérationnel seul, ou les deux ensemble ? Combien de fois a-t-il été consulté pour une décision ?

B6 Sur ton dernier processus critique, as-tu utilisé un organigramme, un diagramme UML, un BPMN, ou autre chose ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Choisir un processus réel de mon organisation qui fait l'objet d'une décision en cours. (Échéance : J+3.)

C2 Dessiner un BPMN macro (5-15 activités, 3-5泳lane) du processus choisi. (Échéance : J+14.)

C3 Identifier 1-3 sous-processus critiques et dessiner leur BPMN détaillé (20-50 activités, exceptions, indicateurs). (Échéance : J+30.)

C4 Tenir un atelier d'une demi-journée avec un opérationnel pour valider conjointement le BPMN. (Échéance : J+45.)

C5 Signer le BPMN conjointement avec l'opérationnel (signature opérationnelle). (Échéance : J+50.)

C6 Diffuser le BPMN signé dans le wiki d'architecture et le comité de gouvernance. (Échéance : J+60.)

L'Architecture des Données : la pyramide DAMA

Architecture d'Entreprise

III — LE CARACTÈRE

« *Data is an asset with unique properties — it is not depleted by use, but can degrade in quality if poorly governed.* » — DAMA, Data Management Body of Knowledge (DMBOK 2.0)

« *Une donnée sans maître est une donnée qui pourrit.* »

Vidéo d'introduction

 Télécharger cette leçon en PDF

La mine de charbon qu'on n'arrive pas à exploiter

En 2019, Marc dirige la Direction Data de GreenTech. Il hérite d'une situation que toute organisation moderne redoute : une **mine de données inexploitée**. GreenTech a quinze ans d'historique de production d'énergie, dix ans de données clients, trois ans de données IoT de capteurs. Au total : 47 téraoctets de données, 217 tables dans 12 bases différentes, des millions de lignes par jour.

Marc a un budget, une équipe, et une demande de la direction : « *exploite ces données pour faire des économies et améliorer la satisfaction client* ». Trois projets pilotes sont lancés en parallèle :

- Un projet de maintenance prédictive des éoliennes (avec les données IoT)
- Un projet de scoring du risque client (avec les données contractuelles)
- Un projet de détection des fraudes (avec les données de paiement)

Les trois projets piétinent. Après quatre mois, aucun n'a livré de valeur en production. L'équipe data passe son temps à comprendre où sont les données, à réconcilier des formats incohérents, à traiter des valeurs manquantes ou aberrantes. La direction commence à douter : « *finalement, l'IA, c'est de la poudre aux yeux* ».

Marc fait alors ce qu'il aurait dû faire dès le début : **regarder les données elles-mêmes**, pas les modèles qu'on veut leur appliquer. Il découvre une réalité brutale.

- Les données de capteurs IoT ont 15 % de valeurs manquantes sur les éoliennes les plus anciennes (capteurs défaillants jamais remplacés).
- Les données clients ont trois identifiants différents selon les applications — un même client peut avoir trois identifiants.
- Les données de paiement sont stockées dans un format propriétaire d'un éditeur, dont l'éditeur a cessé le support en 2018.
- Aucune des tables n'a de **propriétaire** identifié. Personne n'est responsable de la qualité, de la complétude, de la documentation.

Marc pose alors la question qui aurait dû être posée en premier : « **quelle est l'architecture de nos données ?** ». Cette question déclenche un chantier de six mois qui n'a rien à voir avec l'IA, mais qui rend l'IA possible.

C'est ce que cette leçon cherche à vous transmettre : **comment l'architecture des données est le prérequis à toute initiative d'exploitation, et comment la pyramide DAMA donne le cadre de ce travail.**

Voici la pyramide DAMA, qui sert de cadre à toute évaluation de maturité data :

Premier principe — Les données sont un actif, pas un sous-produit

La première erreur conceptuelle consiste à considérer les données comme un sous-produit des applications. Cette erreur rend toute initiative data impossible à coordonner, et condamne les projets à dépendre de la bonne volonté des détenteurs applicatifs.

Le DAMA DMBOK 2.0 est sans ambiguïté sur ce point : « *Data is an asset with unique properties* ». Les données sont un actif de l'organisation, au même titre que les bâtiments, les machines, les brevets. Elles ont trois propriétés uniques :

- **Non-déplétion** — l'usage d'une donnée ne la consomme pas (contrairement à une matière première). Une même donnée peut servir mille usages.
- **Dégradation possible** — la qualité d'une donnée peut se dégrader (obsolescence, valeurs manquantes, formats incompatibles). Contrairement aux matières premières, les données peuvent se dégrader même quand elles sont stockées.
- **Externalités** — l'usage d'une donnée par un service peut affecter un autre service (un changement de format cassant les consommateurs en aval).

Ces trois propriétés justifient que les données soient gérées comme un actif, pas comme un sous-produit. Cela signifie : un propriétaire identifié pour chaque donnée, une politique de qualité documentée, un cycle de revue, un budget dédié.

La contrefaçon classique consiste à dire « nos données sont dans les bases de nos applications, donc elles sont gérées ». Cette formulation cache une réalité : les données ne sont gérées que par accident, au gré des évolutions applicatives, sans vue d'ensemble.

***Et toi ?** Combien de jeux de données critiques de ton organisation ont un **propriétaire identifié** ? Combien ont une politique de qualité documentée ? Combien ont fait l'objet d'une revue annuelle ?*

Deuxième principe — La pyramide DAMA distingue cinq niveaux

Le DAMA DMBOK organise les activités de gestion des données en une pyramide à cinq niveaux. Chaque niveau répond à une question différente. Sauter un niveau rend les niveaux supérieurs intenables.

La pyramide DAMA se présente ainsi, du plus opérationnel au plus stratégique :

1. **Niveau 1 — Données opérationnelles** — les données brutes, telles qu'elles sont créées et consommées par les applications transactionnelles.
2. **Niveau 2 — Données de référence (Master Data)** — les données partagées par plusieurs applications, qui doivent être cohérentes entre elles (clients, produits, fournisseurs).
3. **Niveau 3 — Données historisées (Data Warehouse)** — les données consolidées dans un entrepôt, pour l'analyse historique.

4. **Niveau 4 — Données analytiques (BI / Reporting)** — les indicateurs, dashboards, rapports construits à partir du Data Warehouse.
5. **Niveau 5 — Données décisionnelles (Advanced Analytics / IA)** — les modèles prédictifs, les algorithmes d'optimisation, les recommandations.

La règle d'or est **l'ordre des niveaux** : on ne peut pas faire du niveau 5 (IA) sans le niveau 4 (BI), ni du niveau 4 sans le niveau 3 (historisation), etc. Cette règle est souvent violée dans les organisations qui veulent « faire de l'IA » sans avoir stabilisé les niveaux inférieurs.

Pour GreenTech, Marc a compris le problème après quatre mois de tâtonnement : il essayait de faire du niveau 5 (maintenance prédictive) sans avoir stabilisé les niveaux 1 et 2 (capteurs IoT, données contractuelles). Le chantier de remédiation a commencé par les niveaux les plus bas.

***Et toi ?** Sur quelle strate de la pyramide DAMA ton organisation est-elle mature ? Combien de niveaux sont vraiment outillés et maintenus ? Quel est le niveau le plus faible ?*

Troisième principe — Le Master Data Management résout le problème des données éclatées

Le MDM (Master Data Management) est la pratique qui consiste à désigner un propriétaire et une source de vérité pour chaque donnée partagée. Sans MDM, l'organisation vit dans le chaos des données dupliquées.

Le problème classique du MDM est celui des **données client**. Dans une organisation sans MDM, le même client peut apparaître sous trois identifiants différents selon qu'il a été créé dans le CRM, dans l'ERP, ou dans le système de facturation. Trois identifiants, trois adresses, trois historiques de paiement. Toute analyse client est impossible.

Le MDM résout ce problème en trois étapes :

1. **Identifier les données partagées** — clients, produits, fournisseurs, employés, sites géographiques. Liste courte, 5 à 10 entités en général.
2. **Désigner un propriétaire** — un rôle identifié (pas un service), qui a la responsabilité de la qualité de la donnée.
3. **Établir une source de vérité** — une application ou un référentiel qui contient la version canonique de la donnée. Les autres applications consomment cette version (en lecture) ou la mettent à jour (avec validation).

Pour GreenTech, Marc a commencé le MDM par les **clients** (3 mois) puis les **sites de production** (2 mois). Le résultat : un identifiant unique client, trois applications qui consomment ce même identifiant. Le scoring du risque client, bloqué depuis six mois, est devenu un projet de six semaines.

***Et toi ?** Combien d'applications dans ton organisation contiennent une version de la donnée « client » ? Combien ont des identifiants différents ? Combien ont un propriétaire identifié ?*

Quatrième principe — Le Data Warehouse consolide, mais ne déduplie pas

Le Data Warehouse (DW) est le lieu où les données sont historisées pour l'analyse. Sa fonction n'est pas de dédupliquer, mais de fédérer des sources multiples en respectant leur diversité.

L'école Inmon (Bill Inmon, *Building the Data Warehouse*, 1990) prescrit une approche **top-down** : on conçoit d'abord le

modèle de données global de l'organisation (le « grand sujet »), puis on alimente. L'école Kimball (Ralph Kimball, *The Data Warehouse Toolkit*, 1996) prescrit une approche **bottom-up** : on modélise par sujet d'analyse (« data mart »), et le DW émerge de l'union des data marts.

Les deux écoles ont leurs partisans. Le consensus actuel penche pour une **approche hybride** : un modèle de données global pour la gouvernance (Inmon), des data marts pour l'analyse opérationnelle (Kimball).

La contrefaçon classique consiste à utiliser le DW comme un **dénormaliseur** — on prend les données des applications, on les dénormalise, on stocke, et on espère que ça suffise. Cette approche produit un DW qui ressemble aux données sources, sans valeur ajoutée.

Pour GreenTech, Marc a adopté l'approche hybride : un modèle de données global documenté (3 mois), puis trois data marts pour les trois projets pilotes (maintenance, scoring, fraude). Le DW a émergé progressivement, sans céder à la tentation du « grand DW » qui résoudrait tout.

Et toi ? Ton organisation a-t-elle un DW ? Quelle est son architecture — Inmon, Kimball, ou hybride ? Combien de data marts sont en exploitation active ?

Cinquième principe — La gouvernance des données n'est pas un département, c'est une pratique

La gouvernance des données ne se décrète pas. Elle se pratique au quotidien par les propriétaires de données, soutenus par un petit noyau de coordination.

Le DAMA DMBOK identifie les **rôles clés** de la gouvernance :

- **Data Owner** — la personne responsable de la qualité d'une donnée. C'est un rôle métier, pas un rôle IT. Le propriétaire du « client » est le directeur commercial, pas le DSI.
- **Data Steward** — l'opérationnel qui fait le travail quotidien de qualité des données (dédoublonnage, validation, documentation). C'est un rôle technique ou métier, soutenu par le Data Owner.
- **Data Custodian** — la personne qui gère l'infrastructure technique des données (bases, sauvegardes, sécurité). C'est un rôle IT.
- **Chief Data Officer (CDO)** — le dirigeant qui porte la stratégie data. C'est un rôle exécutif, avec budget et mandat.

La contrefaçon classique consiste à créer un « département Data » qui centralise tous les rôles. Cette organisation rend les Data Owners irresponsables (« c'est le département data qui gère ») et le département data infréquentable. La parade est de **garder la gouvernance distribuée** : les Data Owners restent dans les métiers, soutenus par un petit noyau de CDO + Data Stewards.

Pour GreenTech, Marc a obtenu trois Data Owners nommés (directeur commercial pour les clients, directeur technique pour les sites, directrice financière pour les contrats), un Data Steward dédié, et un mandat CDO rattaché au COMEX. Aucun département data créé. La gouvernance est restée distribuée.

Et toi ? Ton organisation a-t-elle des Data Owners identifiés ? Un CDO ? Combien de Data Stewards à temps plein sur les données critiques ?

Application — Mon plan de remédiation data

À rédiger séance tenante. Cinq sections précises.

1. **Diagnostic** — Sur quelle strate de la pyramide DAMA ton organisation est-elle mature ? Quel est le niveau le plus faible ? Quelle est la donnée partagée la plus éclatée (clients, produits, fournisseurs) ?
2. **Cible** — À 12 mois, quelles strates doivent être opérationnelles ? Quel est le périmètre MDM prioritaire ?
3. **Quick wins** — Trois actions à impact rapide (en dessous de 30 jours) : dédoublonnage d'une table, documentation d'un glossaire, nomination d'un Data Owner.
4. **Programme** — Trois chantiers à 6 mois : MDM client, DW unifié, gouvernance formalisée.
5. **Indicateurs** — Trois KPIs pour mesurer le progrès : taux de dédoublonnage client, complétude des données critiques, nombre de Data Owners nommés.

Ce plan est ton premier livrable d'architecture des données. Il est court, mais il est actionnable.

Cas pratique — Refonte du SI Décisionnel de Mutuelle Plus

Contexte sectoriel : assurance santé, dixième acteur français, 1 000 000 de personnes protégées, 800 collaborateurs. Mutuelle Plus fait face à une crise de son modèle historique : population assurée vieillissante (dépenses de santé en hausse), perte de vitesse face aux concurrents digitaux (résiliations en hausse, 72 heures de délai pour répondre à un adhérent là où le marché est à l'instantanéité).

Diagnostic de l'existant. Le SI est verrouillé par le progiciel **Gest+**. Le Data Warehouse atteint **15 To** de données. Les **batches nocturnes** de calcul de prestations débordent sur les horaires d'ouverture, empêchant les agences d'accéder aux dossiers à jour dès 9h. L'absence de centralisation force les conseillers à utiliser des **tableaux locaux**, brisant la continuité de service lors des mobilités géographiques des adhérents. Impact financier : **150 000 €/an** de pertes directes liées aux incidents de paiement, et hausse continue du taux de résiliation faute de réactivité.

Parties prenantes (matrice de Mendelow) :

Partie prenante	Quadrant	Attente clé
Directeur des Risques	Gérer activement	Conformité RGPD, pilotage solvabilité
DSI	Gérer activement	Désendettement technique Gest+, urbanisation
Directeur Financier	Satisfaire	Retour à l'équilibre des comptes, ROI des projets
Data Analysts	Tenir informé	Accès à une donnée propre pour modéliser
Directeurs d'agences	Tenir informé	Vue 360° du client, fin de la double saisie

Trajectoire de transformation TOGAF ADM :

- **Phase préliminaire + Phase A** : principes d'architecture (partage de l'information, indépendance vis-à-vis de l'éditeur).
- **Transition 1 (Hybridation)** : mise en place d'un bus de données pour extraire les flux de Gest+ en temps réel, soulageant le Data Warehouse et réduisant les erreurs de remboursement.
- **Transition 2 (Convergence)** : déploiement du CRM centralisé et décommissionnement progressif des silos « tableaux » en agence.
- **Architecture cible** : SI orienté services (SOA) facilitant l'intégration de garanties spécifiques pour les agents territoriaux.

Livrables attendus de l'apprenant :

1. Une note de Vision (Phase A) positionnant la cible data à 24 mois.
2. Un schéma logique de données cible (Phase C) avec MDM client.
3. Une roadmap de migration (Phases E/F) structurée en trois vagues.
4. Un Business Case incluant la réduction du churn estimée (3 % de churn évité = ~2,4 M€ de CA stabilisé).

Pièges classiques à éviter :

- Sous-estimer la latence lors du transfert des 15 To vers le nouveau référentiel.
- Oublier la conformité RGPD (projet prioritaire identifié dès le départ).
- Négliger la moyenne d'âge des équipes (43 ans) qui nécessite un plan d'accompagnement au changement IT.

Critères d'évaluation : la note est validée si l'apprenant (1) chiffre l'économie attendue, (2) propose une trajectoire incrémentale, (3) identifie les trois risques majeurs et leurs parades.

Encadré Au lecteur / À l'animateur

Question d'ouverture : « Combien de jeux de données critiques ont un propriétaire identifié aujourd'hui dans ton organisation ? »

Profils à observer :

- **Le data scientist qui veut faire de l'IA sans avoir stabilisé les niveaux 1-2.** Vole au-dessus de la pyramide DAMA. *Question :* « Quel est ton taux de valeurs manquantes sur la donnée critique ? »
- **Le MDM-maniaque qui veut tout consolider.** Vise le « grand MDM unifié » qui résoudra tout. *Question :* « Combien de données partagées sont dans ton périmètre ? »
- **Le data Owner par défaut.** Désigne le DSI comme propriétaire de toutes les données, ce qui rend personne responsable. *Question :* « Qui est nommé responsable de la donnée client ? »
- **L'isolé qui documente seul.** Produit un glossaire de 300 termes que personne ne consulte. *Question :* « Combien de fois ton glossaire a-t-il été consulté pour une décision ? »

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les outils (Data Lake, Snowflake, BigQuery). Rappeler que l'architecture des données est d'abord une question de gouvernance.
- Ne pas confondre Data Lake et Data Warehouse. Le Data Lake stocke des données brutes ; le Data Warehouse les historise avec un modèle.

Ce que la leçon 8 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- **Un diagnostic** de la maturité data de ton organisation selon la pyramide DAMA.
- **Un périmètre MDM prioritaire :** 1 à 3 données partagées, avec Data Owner nommés.
- **Trois quick wins data** à lancer dans les 30 jours.

Sources de cette leçon : DAMA International, *Data Management Body of Knowledge (DMBOK 2.0)* · Bill Inmon, *Building the Data Warehouse* (1990) · Ralph Kimball, *The Data Warehouse Toolkit* (1996) · Cas GreenTech (anonymisé)

Leçon suivante : **L8 — L'Architecture Applicative : la cartographie des systèmes.**

Quiz — 18 questions pour valider votre compréhension

Trois series de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Quelles sont les trois propriétés uniques des données selon le DAMA DMBOK ? Pourquoi justifient-elles qu'on les traite comme un actif ?

A2 Quels sont les cinq niveaux de la pyramide DAMA ? Pourquoi doit-on respecter l'ordre des niveaux ?

A3 Qu'est-ce que le Master Data Management (MDM) ? Quelles sont les trois étapes pour le mettre en place ?

A4 Quelle est la différence entre l'approche Inmon et l'approche Kimball pour le Data Warehouse ? Pourquoi l'approche hybride est-elle recommandée ?

A5 Quels sont les quatre rôles clés de la gouvernance des données (Data Owner, Steward, Custodian, CDO) ? Quelle est la contrefaçon à éviter ?

A6 Pourquoi un projet d'IA (niveau 5) sans Data Warehouse (niveau 3) ni MDM (niveau 2) est-il condamné à piétiner ?

B — AUTO-EVALUATION

B1 Combien de jeux de données critiques de ton organisation ont un propriétaire identifié ? Une politique de qualité documentée ? Une revue annuelle ?

B2 Sur quelle strate de la pyramide DAMA ton organisation est-elle mature ? Quel est le niveau le plus faible ?

B3 Combien d'applications dans ton organisation contiennent une version de la donnée « client » ? Combien ont des identifiants différents ? Combien ont un propriétaire identifié ?

B4 Ton organisation a-t-elle un Data Warehouse ? Quelle est son architecture — Inmon, Kimball, ou hybride ? Combien de data marts sont en exploitation ?

B5 Ton organisation a-t-elle des Data Owners identifiés ? Un CDO ? Combien de Data Stewards à temps plein ?

B6 Combien de jeux de données critiques ont un propriétaire identifié aujourd'hui dans ton organisation ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Diagnostiquer la maturité data de mon organisation selon la pyramide DAMA (5 niveaux, score 1-5). (Échéance : J+7.)

C2 Définir le périmètre MDM prioritaire : 1 à 3 données partagées avec Data Owners nommés. (Échéance : J+14.)

C3 Lancer trois quick wins data : dédoublonnage d'une table, documentation d'un glossaire, nomination d'un Data Owner. (Échéance : J+30.)

C4 Lancer un chantier MDM client sur 3 mois : identifier le propriétaire, établir la source de vérité, réconcilier les applications. (Échéance : J+90.)

C5 Définir trois KPIs data : taux de dédoublonnage client, complétude des données critiques, nombre de Data Owners nommés. (Échéance : J+45.)

C6 Documenter la gouvernance data : matrice RACI des données critiques, revue trimestrielle. (Échéance : J+60.)

L'Architecture Applicative : le découplage des systèmes

Architecture d'Entreprise

III — LE CARACTÈRE

« Conway's Law strikes back: organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations. » — Melvin Conway, *How Do Committees Invent?* (1968)

« Un catalogue d'APIs n'est pas une architecture applicative. »

Vidéo d'introduction

 Télécharger cette leçon en PDF

La tour de spaghetti qui a failli couler Industrias Yara

En 2017, Industrias Yara, ETI industrielle de 2 800 salariés, connaît une croissance rapide après l'acquisition de deux concurrents. La DSI, jusqu'alors organisée autour de trois applications principales (un ERP maison, un CRM maison, une GED maison), se retrouve à devoir intégrer six nouvelles applications métiers issues des acquisitions — chacune avec ses propres connecteurs, ses propres formats de données, ses propres conventions de nommage.

Au bout de dix-huit mois, l'architecture applicative ressemble à un plat de spaghetti. Chaque application parle à cinq autres applications, parfois via des flux de fichiers CSV déposés sur un partage réseau, parfois via des web services dont personne ne maintient la documentation, parfois via des triggers SQL dont seuls deux développeurs connaissent l'existence. Quand un développeur part à la retraite, deux projets d'intégration tombent en panne.

Camille, l'architecte applicatif recrutée en urgence, fait trois constats sans complaisance :

1. **Il n'y a pas d'architecture applicative, seulement un historique.** Les choix applicatifs ont été faits par sédimentation, en réaction à des urgences, sans vision d'ensemble. Le résultat est cohérent avec l'histoire de l'organisation, mais pas avec ses besoins présents.
2. **Personne ne sait ce qui dépend de quoi.** La cartographie applicative est obsolète depuis cinq ans. Personne n'a une vision complète des interfaces entre systèmes. Quand on demande « qui appelle l'API X ? », les réponses sont incomplètes ou contradictoires.
3. **Le pattern « Just a Bunch Of Web Services » (JBOWS) est partout.** Chaque équipe a créé ses propres APIs sans gouvernance transverse. Résultat : des APIs redondantes (cinq APIs pour la donnée client), des APIs sans contrat d'interface, des APIs orphelines (personne ne les utilise mais personne ne les décommissionne).

Camille passe six mois à reconstruire une **cartographie applicative** à jour, à établir des **contrats d'interface** pour les APIs critiques, et à mettre en place un **catalogue de services** qui distingue les APIs techniques des services métier. Le chantier n'élimine pas la complexité — il la rend visible et gérable.

C'est ce que cette leçon cherche à vous transmettre : **comment passer d'une tour de spaghetti à une architecture**

applicative gouvernée, sans naïveté sur ce que ce travail peut (et ne peut pas) accomplir.

La cartographie applicative d'Industrias Yara, organisée par niveaux de criticité, illustre comment une organisation peut passer d'une vue en spaghetti à une vue en strates gouvernables :

Premier principe — JBOWS n'est pas de l'architecture, c'est un anti-pattern

Le terme « Just a Bunch Of Web Services » désigne une collection d'APIs sans gouvernance transverse. Cette collection n'est pas une architecture — c'est un anti-pattern.

Sam Newman (*Building Microservices*, 2015) a popularisé l'expression JBOWS pour dénoncer la fausse sophistication de certaines architectures distribuées. Une collection de 200 APIs mal gouvernées n'est pas plus « moderne » qu'un monolithe bien structuré. Elle est souvent **moins** performante, **moins** fiable, et **plus** coûteuse à maintenir.

Les signes distinctifs d'un système JBOWS :

- Pas de contrat d'interface explicite — chaque API documente ce qu'elle veut
- Pas de propriétaire identifié pour chaque API — quand l'API casse, personne ne réagit
- Pas de versionning cohérent — les breaking changes arrivent sans prévenir
- Pas de mesure d'usage — on ne sait pas si elle est utilisée par 1 ou 100 clients
- Pas de politique de décommissionnement — les APIs mortes restent actives indéfiniment

La contrefaçon classique consiste à croire que « passer aux microservices » résoudra ces problèmes. En réalité, **les microservices sans gouvernance ne font que multiplier le problème par N**. La parade est de mettre en place la gouvernance **avant** la multiplication des services.

Pour Industrias Yara, Camille a démarré par un **inventaire honnête** : 187 APIs identifiées, dont 67 sans propriétaire, 23 sans documentation, 8 sans aucun appelant connu sur les 90 derniers jours. Première action : décommissionner les 8 APIs orphelines. Résultat : moins de surface à maintenir, démonstration tangible que la gouvernance fonctionne.

***Et toi ?** Combien d'APIs ton organisation expose-t-elle ? Combien ont un propriétaire identifié ? Combien sont documentées ? Combien sont utilisées par au moins un client en production ?*

Deuxième principe — La cartographie applicative est vivante, pas commémorative

Une cartographie applicative qui n'est pas mise à jour régulièrement devient un mensonge documentaire — pire qu'une absence de cartographie, car elle induit les décideurs en erreur.

L'Open Group recommande que la cartographie applicative soit **maintenue activement**, avec :

- **Un propriétaire identifié** — un architecte applicatif (souvent à temps plein) qui maintient la cartographie comme une activité continue, pas comme un livrable de projet
- **Une source de vérité unique** — un outil partagé (ArchiMate, LeanIX, Mega HopEx) où la cartographie est éditée
- **Un cycle de revue** — trimestriel pour les applications critiques, annuel pour les autres
- **Des indicateurs d'usage** — chaque application est documentée avec son taux d'utilisation, son coût de run, son risque de fin de support éditeur

La contrefaçon classique consiste à produire une cartographie de 87 applications dans un PowerPoint de 200 pages, jamais mise à jour, oubliée sur un sharepoint. Cette cartographie-obélisque n'a aucune valeur opérationnelle. Elle a même une valeur **négative** : les décideurs croient qu'elle est à jour et prennent des décisions sur la base d'informations fausses.

Pour Industrias Yara, Camille a mis en place un **outil léger** : un tableur partagé entre cinq architectes applicatifs, avec une cellule « dernière mise à jour » par ligne. Le cycle de revue est mensuel pour les 20 applications critiques, trimestriel pour les autres. La cartographie a 90 % des informations vérifiées dans les 90 jours, contre 30 % avant la réforme.

***Et toi ?** À quand remonte la dernière mise à jour de ta cartographie applicative ? Quel pourcentage des applications sont documentées à jour ? Quel outil utilises-tu ?*

Troisième principe — Le contrat d'interface est un acte juridique, pas technique

Une API sans contrat d'interface est une API sans propriétaire. Le contrat n'est pas un fichier WSDL ou OpenAPI — c'est un engagement entre l'équipe qui produit l'API et celles qui la consomment.

Eric Evans (*Domain-Driven Design*, 2003) a popularisé la notion de **Bounded Context** : chaque service métier a une frontière claire, avec un langage et un contrat explicites. Cette notion est au cœur d'une architecture applicative mature.

Un contrat d'interface document devrait contenir au minimum :

- **Le périmètre fonctionnel** — quels cas d'usage l'API couvre, et (surtout) lesquels elle **ne couvre pas**
- **Le propriétaire** — un nom, une équipe, une adresse de support
- **La SLA** — temps de réponse attendu, taux de disponibilité, temps de résolution d'incident
- **Le contrat de données** — le schéma des entrées et sorties, documenté (OpenAPI, Protobuf, Avro)
- **La politique de versionning** — comment les breaking changes sont annoncés, combien de versions coexistent
- **La politique de dépréciation** — combien de temps une ancienne version reste supportée après une nouvelle

La contrefaçon classique consiste à publier une spécification OpenAPI et à croire que le contrat est signé. Sans les engagements de SLA, de versionning, et de dépréciation, le contrat reste incomplet — et l'équipe consommatrice ne sait pas ce qu'elle peut attendre.

Pour Industrias Yara, Camille a imposé un **template de contrat** de deux pages, signé conjointement par l'équipe qui produit l'API et un représentant des consommateurs. La signature n'est pas un acte bureaucratique — c'est un engagement de support.

***Et toi ?** Combien d'APIs dans ton organisation ont un contrat complet (périmètre, SLA, versionning, dépréciation) ? Combien n'ont que la spécification technique ?*

Quatrième principe — L'ERP est un système complexe, pas une boîte noire

L'ERP est le système qui concentre le plus de complexité — et le plus de risques. Le traiter comme une boîte noire qu'on « configure » est une illusion. Le comprendre comme un système complexe qu'on doit maîtriser est une nécessité.

L'ERP (SAP, Oracle, Microsoft Dynamics, etc.) cumule quatre caractéristiques qui en font un objet d'architecture à part entière :

1. **Il est critique** — la majorité des processus métiers passent par lui. Un incident ERP peut paralyser l'organisation.
2. **Il est paramétrique, pas configurable** — chaque paramétrage a des implications en cascade. « Configurer » un ERP sans comprendre son modèle de données produit des comportements imprévus.
3. **Il évolue avec l'éditeur** — les montées de version imposent des refontes de paramétrage. L'ERP n'est jamais stable.
4. **Il porte la dette technique** — les paramétrages successifs accumulent de la dette. Un ERP « vieux » de 15 ans peut être plus complexe à maîtriser qu'un ERP récent mal paramétré.

La contrefaçon classique consiste à externaliser l'ERP à un intégrateur et à perdre la maîtrise interne du système. L'organisation devient alors dépendante de l'intégrateur pour chaque évolution, et la dette technique s'accumule sans visibilité.

La parade est de maintenir une **équipe interne ERP** (même petite), avec :

- Un **architecte ERP** qui comprend le modèle de données
- Un **responsable de la maîtrise d'ouvrage** qui porte les besoins métiers
- Un **processus de revue** des paramétrages critiques (workflow, comptabilité, paie)

Pour Industrias Yara, Camille a créé un poste d'architecte ERP dédié — une décision qui a coûté 90 k€ par an et a évité des millions d'euros d'incidents et de projets mal cadrés.

Et toi ? Combien de personnes dans ton organisation maîtrisent le modèle de données de l'ERP ? Combien de paramétrages critiques ont été documentés ? Combien de fois l'intégrateur a-t-il été appelé pour un incident causé par un paramétrage obscur ?

Cinquième principe — Un catalogue de services n'est pas une liste d'APIs

Le catalogue de services est un outil de gouvernance métier, pas un annuaire technique. Il sert à répondre à la question « quel service métier est exposé par quelle application, sous quelle responsabilité ? » — pas « voici nos 200 URLs ».

Un catalogue de services mature contient, pour chaque service :

- **Le nom métier** — un nom compréhensible par un opérationnel (« service de souscription client »)
- **Le périmètre fonctionnel** — ce que le service fait, en langage métier
- **L'application qui l'expose** — nom de l'application, éditeur, version
- **L'API technique** — URL, contrat, version
- **Le propriétaire métier** — la direction responsable du service
- **Le propriétaire technique** — l'équipe qui maintient l'API
- **Les SLA** — engagements de performance et de disponibilité
- **Les consommateurs** — qui appelle ce service, dans quel but

La contrefaçon classique consiste à publier une **liste d'APIs** (URL + description technique) en la qualifiant de catalogue. Cette liste ne sert qu'aux développeurs, pas aux opérationnels ni aux décideurs. Le catalogue de services sert à **rendre l'architecture applicative discutable au niveau métier**.

Pour Industrias Yara, Camille a mis en place un catalogue à deux niveaux : un **catalogue métier** (15 services, lisible par un COMEX) et un **catalogue technique** (187 APIs, réservé aux architectes). Les deux catalogues sont liés, mais leur

public est différent.

***Et toi ?** Combien de services métier sont exposés par ton SI ? Combien ont un propriétaire métier identifié ? Combien ont une SLA publiée ?*

Application — Ma cartographie applicative

À produire séance tenante. Cinq sections précises.

1. **Inventaire** — Liste exhaustive des applications de ton organisation (nom, éditeur, version, date de mise en production).
2. **Critique** — Pour chaque application, note de criticité (1 à 5), avec la justification.
3. **Contrats** — Pour les 10 applications les plus critiques, identification des 5 principaux contrats d'interface (fournisseur ou consommateur).
4. **APIs orphelines** — Liste des APIs sans propriétaire, sans usage, ou sans documentation. Cible de décommissionnement à 90 jours.
5. **Catalogue métier** — Liste des 10 à 15 services métier exposés, avec leur propriétaire métier nommé.

|> Cette cartographie est ton premier livrable d'architecture applicative. Elle est la base de toute gouvernance ultérieure.

Module complémentaire — L'API-first comme principe architectural

Dans un SI moderne, l'application n'est plus un silo. Elle est un **point d'entrée** dans un écosystème de services. La question n'est plus « quelle application possède cette donnée ? » mais « **quelle API expose cette donnée, et à quelles conditions ?** ».

Le principe API-first

L'**API-first** est une approche où les **interfaces** (APIs) sont conçues *avant* les implémentations. Ce n'est pas un choix technique : c'est un **principe architectural** qui transforme la gouvernance applicative.

Trois implications concrètes :

1. **L'API est un produit, pas un canal.** Elle a un propriétaire nommé, un cycle de vie, une documentation, une SLA de latence et de disponibilité, un versioning, une politique de dépréciation.
2. **Le contrat d'interface précède l'implémentation.** Les consommateurs peuvent s'intégrer avant que l'implémentation ne soit prête (par mock, par stub), ce qui parallélise le développement et clarifie les attentes.
3. **L'API est l'unique vecteur d'accès aux données.** Aucun accès direct aux bases de données (SQL direct) n'est toléré. Cette règle, héritée du cas Retail (L4 — Schéma Directeur Omnicanal), est un garde-fou de **sécurité** et de **cohérence**.

Le « Single Source of Truth » et la gouvernance des données

L'API-first rejoint le principe du **Single Source of Truth** (SSOT) : pour chaque donnée critique, il existe **une seule API de référence** qui fait foi. Toutes les autres sources (réplicas, caches, bases de shadow IT) doivent se synchroniser

depuis cette API. Sans ce principe, l'organisation cumule les versions divergentes de la même information — et les erreurs qui en découlent.

Chez Mutuelle Plus, l'absence de SSOT sur la donnée client (Gest+ + 50 tableurs Excel en agence) génère les doublons qui plombent le délai de remboursement. La mise en place d'une API « Client » comme unique source de vérité est une **condition préalable** à la refonte du SI Décisionnel (voir L7 — Architecture des Données, cas pratique Mutuelle Plus).

Trois erreurs de gouvernance API-first à éviter

1. « **API-flood** » : exposer une API pour chaque besoin sans regroupement logique. Le consommateur se retrouve à intégrer 50 APIs pour un seul processus métier. La règle : une **API métier** doit couvrir un **domaine fonctionnel** (ex. : API « Contrats »), pas une seule table.
2. **API sans propriétaire** : l'API est publiée mais personne n'est nommé responsable. Quand un incident survient, personne ne répond. Solution : un **registre des APIs** avec, pour chacune, un Product Owner métier, un Tech Lead, et un SLA documenté.
3. **Pas de politique de dépréciation** : les APIs s'accumulent, certaines ne sont plus consommées mais continuent à coûter (maintenance, hébergement, monitoring). Planifier une revue trimestrielle des APIs sous-utilisées et décommissionner.

L'outillage minimum

Une gouvernance API-first efficace repose sur quatre composants :

- **Un catalogue d'APIs** (Backstage, Stoplight, ReadMe) qui sert de portail unique aux développeurs.
- **Une API Gateway** qui centralise l'authentification, le throttling, le monitoring, et la journalisation.
- **Des standards de description** (OpenAPI / Swagger pour REST, AsyncAPI pour les flux événementiels).
- **Un registre de schémas** qui garantit la compatibilité ascendante et gère les évolutions de contrats.

Synthèse

L'API-first n'est pas un projet technique. C'est un **changement de paradigme** : l'organisation passe d'une logique de silos applicatifs à une logique de **services réutilisables**. Cela nécessite un investissement initial (catalogue, gateway, formation) et une discipline continue (revues, dépréciations). Le ROI se mesure en **vitesse d'intégration** (nouveau partenaire en moins de quatre semaines, comme dans le cas Retail) et en **réduction des incohérences** (zéro doublon dans la donnée client, comme dans le cas Mutuelle Plus).

Encadré Au lecteur / À l'animateur

Question d'ouverture : « *Combien d'APIs ton organisation expose-t-elle aujourd'hui ? Combien ont un propriétaire identifié ?* »

Profils à observer :

- **Le microservice-maniaque**. Croit que la multiplication des APIs résoudra les problèmes. *Question* : « *Combien d'APIs as-tu sans propriétaire ?* »
- **Le collectionneur de cartographies**. Produit une cartographie PowerPoint de 200 pages, jamais mise à jour. *Question* : « *À quand remonte la dernière mise à jour ?* »
- **Le dépendant de l'intégrateur**. A externalisé l'ERP et perdu la maîtrise interne. *Question* : « *Combien de personnes maîtrisent le modèle de données ?* »
- **L'API-phobe**. Refuse de documenter ses APIs par peur de l'engagement. *Question* : « *Combien de consommateurs ont signé un contrat avec toi ?* »

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les outils (Kubernetes, Istio, etc.). Rappeler que l'architecture applicative est d'abord une question de gouvernance.
- Ne pas confondre API et service. Une API est une implémentation technique ; un service est un engagement métier.

Ce que la leçon 9 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- **Une cartographie applicative** à jour, avec criticité notée.
- **10 contrats d'interface** documentés pour les applications critiques.
- **Un catalogue métier** de 10 à 15 services, avec propriétaires nommés.

Sources de cette leçon : TOGAF Series Guide to Microservices Architecture (G243) · ArchiMate 3.2 Specification · Eric Evans, *Domain-Driven Design* (2003) · Sam Newman, *Building Microservices* (2015) · Cas Industrias Yara (anonymisé)

Leçon suivante : **L9 — L'Architecture Technique : les services de plate-forme.**

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Qu'est-ce que l'anti-pattern JBOWS ? Pourquoi la multiplication des APIs sans gouvernance est-elle contre-productive ?

A2 Pourquoi une cartographie applicative doit-elle être vivante (maintenue activement) plutôt qu'un livrable de projet commémoratif ?

A3 Qu'est-ce qu'un contrat d'interface complet (périmètre, SLA, versionning, dépréciation) ? Pourquoi la spécification technique OpenAPI seule ne suffit-elle pas ?

A4 Quelles sont les quatre caractéristiques qui font de l'ERP un objet d'architecture à part entière ? Pourquoi faut-il maintenir une équipe interne ?

A5 Quelle est la différence entre un catalogue de services et une liste d'APIs ? À quel public chacun s'adresse-t-il ?

A6 Qu'est-ce que la loi de Conway ? En quoi est-elle pertinente pour l'architecture applicative ?

B — AUTO-EVALUATION

B1 Combien d'APIs ton organisation expose-t-elle ? Combien ont un propriétaire identifié ? Combien sont documentées ? Combien sont utilisées par au moins un client en production ?

B2 À quand remonte la dernière mise à jour de ta cartographie applicative ? Quel pourcentage des applications sont documentées à jour ? Quel outil utilises-tu ?

B3 Combien d'APIs dans ton organisation ont un contrat complet (périmètre, SLA, versionning, dépréciation) ? Combien n'ont que la spécification technique ?

B4 Combien de personnes dans ton organisation maîtrisent le modèle de données de l'ERP ? Combien de paramètres critiques ont été documentés ?

B5 Combien de services métier sont exposés par ton SI ? Combien ont un propriétaire métier identifié ? Combien ont une SLA publiée ?

B6 Sur ton dernier projet d'architecture, as-tu rencontré une tour de spaghetti ? Combien d'APIs as-tu identifiées sans propriétaire ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Produire un inventaire exhaustif des applications de mon organisation (nom, éditeur, version, date de mise en production). (Échéance : J+14.)

C2 Attribuer une note de criticité (1 à 5) à chaque application avec justification. (Échéance : J+21.)

C3 Identifier les 10 applications les plus critiques et documenter leurs 5 principaux contrats d'interface (fournisseur ou consommateur). (Échéance : J+45.)

C4 Lister les APIs orphelines (sans propriétaire, sans usage, ou sans documentation) et planifier leur décommissionnement. (Échéance : J+30.)

C5 Construire un catalogue métier de 10 à 15 services, avec leur propriétaire métier nommé. (Échéance : J+60.)

C6 Mettre en place un cycle de revue trimestriel pour les 20 applications les plus critiques. (Échéance : J+90.)

L'Architecture Technique : les services de plate-forme

Architecture d'Entreprise

III — LE CARACTÈRE

« The Technical Reference Model (TRM) is a reference model that provides a vocabulary and a structure for describing technical components. » — The Open Group, TOGAF Series Guide — TRM (G229)

« Une bonne architecture technique n'est pas celle qui choisit les bons outils. C'est celle qui peut en changer sans tout réécrire. »

Vidéo d'introduction

📄 Télécharger cette leçon en PDF

Le datacenter qu'on n'arrivait plus à éteindre

En 2020, Mapify — éditeur SaaS de cartographie pour les collectivités — fait face à une croissance subite. 200 % d'augmentation du trafic en 18 mois, liée à une série de contrats publics. La plate-forme, conçue en 2018 pour 10 000 utilisateurs simultanés, est dimensionnée pour 50 000 — mais les incidents se multiplient. Trois coupures majeures en six mois, dont une de 14 heures qui a fait la une de la presse spécialisée.

L'équipe technique, brillante mais submergée, a fait ce que beaucoup d'équipes brillantes font dans cette situation : **empiler les solutions ponctuelles**. Load balancer supplémentaire devant le load balancer existant, base de données répliquée à la hâte sur un autre datacenter, services de cache improvisés. Six mois plus tard, personne — y compris les architectes — n'a une vision claire de l'architecture technique en production. La documentation est obsolète. Les diagrammes sont des souvenirs.

C'est à ce moment qu'Antoine, l'architecte technique, est recruté. Son premier acte est contre-intuitif : **ne pas changer l'infrastructure existante**. Son deuxième acte l'est tout autant : **représenter l'architecture technique telle qu'elle est vraiment, pas telle qu'elle était censée être**.

Antoine passe un mois à interviewer les équipes, à examiner les configurations réelles (pas celles documentées), à reconstituer la **stack technique effective** — serveurs, réseaux, stockage, middleware, services d'observabilité. Le résultat est un **Technical Reference Model (TRM)** adapté à Mapify : un modèle de référence qui décrit ce qui existe, avec ses forces et ses fragilités.

Une fois le TRM posé, Antoine peut alors distinguer : - **Les services de plate-forme durables** — les composants qu'il faut investir, à stabiliser, à gouverner - **Les composants à risque** — ceux qui marchent par chance ou par héroïsme individuel, et qu'il faut reprendre - **Les candidats au décommissionnement** — ceux qui ne servent plus, mais continuent à coûter

Le TRM est la première étape d'une architecture technique qui **décrit l'existant** avant de prescrire la cible.

C'est ce que cette leçon cherche à vous transmettre : **comment le TRM rend visible l'architecture technique effective, et comment il sert de base à toute décision de plate-forme**.

Le TRM de Mapify illustre les sept catégories de services techniques que toute organisation doit cartographier — avec, au milieu, la plate-forme qui les relie :

Premier principe — Le TRM décrit ce qui est, pas ce qu'on voudrait

Le TRM (Technical Reference Model) est un modèle de référence qui décrit la stack technique effective d'une organisation — pas la stack idéale. Cette distinction est cruciale : sans description honnête de l'existant, toute décision de cible est construite sur du sable.

L'Open Group publie un TRM générique qui identifie les grandes catégories de services techniques :

- **Services applicatifs** — l'exécution des processus métier
- **Services de données** — la persistance, l'indexation, la réplication
- **Services d'infrastructure** — serveurs, stockage, réseau, virtualisation
- **Services de plate-forme** — middleware, orchestration, service mesh
- **Services de sécurité** — identité, authentification, chiffrement, audit
- **Services d'observabilité** — monitoring, logging, tracing, alerting
- **Services de déploiement** — CI/CD, infrastructure as code, automatisation

Le TRM générique est utile comme **grille de lecture** — il permet de cartographier l'existant dans chaque catégorie. Mais le TRM **spécifique** à l'organisation est ce qui compte : il décrit les produits concrets utilisés (AWS EC2, PostgreSQL 15, Istio 1.20, etc.), avec leur criticité, leur dette, leur risque de fin de support.

La contrefaçon classique consiste à dessiner un TRM idéal (Kubernetes partout, microservices, observabilité full-stack) sans décrire l'existant. Cette fiction rend les slides impressionnantes mais les décisions intenable.

***Et toi ?** As-tu un TRM documenté pour ton organisation ? Décrit-il l'existant ou la cible ? Quelle est la dernière date de mise à jour ?*

Deuxième principe — La plate-forme abstraite rend la migration possible

Une plate-forme abstraite n'est pas une plate-forme Kubernetes ou AWS — c'est une couche d'abstraction entre les applications et l'infrastructure sous-jacente. Cette abstraction permet de migrer sans réécrire.

L'Open Group recommande que les applications soient déployées sur des **plate-formes abstraites** — des couches logicielles qui masquent les détails de l'infrastructure sous-jacente. Les applications consomment des services de plate-forme (calcul, stockage, réseau) sans connaître les détails du cloud provider.

Trois niveaux d'abstraction courants :

- **Niveau 1 — Aucune abstraction** — les applications connaissent l'infrastructure (connexion directe à RDS, à S3, à un load balancer spécifique)
- **Niveau 2 — Abstraction partielle** — les applications utilisent des API génériques (calcul, stockage, base de données) mappées vers une implémentation spécifique
- **Niveau 3 — Abstraction totale** — les applications utilisent des services de plate-forme (Kubernetes, service mesh, PaaS) qui sont eux-mêmes portables entre clouds

Pour Mapify, Antoine a constaté que les applications étaient au **Niveau 1** : elles connaissaient AWS par leur code (variables d'environnement, SDK spécifiques). La migration vers un autre cloud provider aurait nécessité de réécrire chaque application. Le chantier d'abstraction vers le Niveau 2 (via Terraform et des API génériques) a été lancé sur 18 mois.

***Et toi ?** Ton organisation est-elle au Niveau 1, 2 ou 3 d'abstraction ? Combien de temps faudrait-il pour migrer vers un autre cloud provider aujourd'hui ?*

Troisième principe — Le middleware est un choix structurant, pas une commodité

Le middleware (bus de messages, ESB, API gateway, orchestrateur) structure la circulation de l'information dans le SI. Le choisir sans vision d'ensemble produit des redondances, des conflits, et des coûts cachés.

Le middleware est souvent vu comme une couche technique « sous le capot ». En réalité, c'est une couche structurante qui conditionne la manière dont les applications communiquent. Trois questions doivent être posées avant tout choix de middleware :

- **Le middleware standardise-t-il ou contraint-il ?** Un bus de messages standardise le format des échanges ; un ESB contraint les applications à un modèle spécifique.
- **Le middleware est-il une cible de lock-in ?** Kafka, RabbitMQ, ServiceMix, Mule — chacun verrouille différemment l'organisation.
- **Le middleware permet-il l'observabilité ?** Sans tracing distribué, sans monitoring des flux, le middleware devient une boîte noire.

Pour Mapify, Antoine a constaté que trois middlewares cohabitaient : un RabbitMQ pour les flux asynchrones, un Kong API Gateway pour les APIs REST, un Sidekiq pour les jobs en arrière-plan. Trois technologies, trois modèles de supervision, trois compétences distinctes dans l'équipe. La rationalisation vers un duo (Kafka + Kong) a permis de réduire la complexité opérationnelle de moitié.

***Et toi ?** Combien de middlewares cohabitent dans ton organisation ? Sont-ils gouvernés par une équipe transverse ou par chaque application ? Combien de compétences différentes cela exige-t-il ?*

Quatrième principe — L'observabilité n'est pas un luxe, c'est une infrastructure critique

On ne peut pas exploiter ce qu'on ne peut pas observer. L'observabilité (logs, métriques, traces) est une infrastructure de première classe, pas une couche ajoutée après coup.

L'observabilité est ce qui permet de répondre à trois questions en production :

- **Que se passe-t-il ?** (métriques)
- **Pourquoi cela se passe-t-il ?** (logs)
- **Comment les composants interagissent-ils ?** (traces)

Sans observabilité, les incidents se résolvent à l'aveugle, par essais-erreurs, en consultant les développeurs qui étaient de garde. C'est coûteux, lent, et épuisant pour les équipes.

La contrefaçon classique consiste à acheter un outil d'observabilité (Datadog, New Relic, Grafana) en croyant que le problème est réglé. L'observabilité n'est pas un outil — c'est une **discipline** : instrumenter le code, centraliser les logs, corréler les traces, définir des SLOs.

Pour Mapify, Antoine a lancé un chantier d'observabilité sur 6 mois : OpenTelemetry pour l'instrumentation, Grafana pour les dashboards, PagerDuty pour l'alerting. Le résultat : temps moyen de résolution d'incident divisé par 3, sommeil des équipes de garde préservé.

***Et toi ?** Combien de temps faut-il à ton équipe pour diagnostiquer un incident en production ? Combien d'outils d'observabilité cohabitent ? As-tu des SLOs définis pour les services critiques ?*

Cinquième principe — La sécurité est une propriété architecturale, pas une couche

La sécurité ne s'ajoute pas à la fin — elle se conçoit dès l'architecture. Une architecture qui ne pense pas la sécurité dès le départ produit des systèmes qu'on doit « sécuriser » après coup, généralement avec des résultats médiocres.

L'Open Group identifie cinq domaines de sécurité à traiter en architecture :

- **Identité et authentification** — qui est l'utilisateur ? comment prouve-t-il son identité ?
- **Autorisation** — que peut-il faire ? selon quelles règles ?
- **Chiffrement** — quelles données sont chiffrées, au repos et, en transit ?
- **Audit** — qui a fait quoi, quand ?
- **Conformité** — RGPD, PCI-DSS, ISO 27001 — quelles contraintes s'appliquent ?

La contrefaçon classique consiste à ajouter la sécurité comme une couche (firewall, antivirus, SIEM) sans revoir l'architecture. Cette approche produit des systèmes où la sécurité est contournée par des pratiques officieuses — partage de credentials, services non documentés, accès administrateur trop larges.

Pour Mapify, Antoine a institué un **Security by Design** : chaque nouvelle application passe par une revue d'architecture sécurité avant d'être lancée. Le coût initial est plus élevé (deux à quatre jours par application), mais le coût total est inférieur (pas de remédiation après incident).

***Et toi ?** Combien de revues d'architecture sécurité sont conduites chaque année dans ton organisation ? Combien d'applications sont concernées par chaque revue ? Quel est le délai moyen entre une vulnérabilité détectée et sa remédiation ?*

Application — Mon TRM

À produire séance tenante. Sept sections précises.

1. **Services applicatifs** — Liste des technologies utilisées (langages, frameworks, runtimes) avec criticité.
2. **Services de données** — Liste des SGBD, data warehouses, data lakes avec criticité.
3. **Services d'infrastructure** — Cloud provider, on-premise, virtualisation, conteneurs avec criticité.
4. **Services de plate-forme** — Middleware, orchestrateurs, service mesh avec criticité.
5. **Services de sécurité** — IdP, IAM, secrets management, pare-feu avec criticité.

6. **Services d'observabilité** — Logs, métriques, traces, alerting avec criticité.
7. **Services de déploiement** — CI/CD, IaC, automatisation avec criticité.

Ce TRM est ton premier livrable d'architecture technique. Il décrit l'existant avec ses forces et ses fragilités.

Encadré Au lecteur / À l'animateur

Question d'ouverture : « Décris en 5 minutes la stack technique de ton organisation. Combien de temps as-tu mis pour la documenter ? »

Profils à observer :

- **Le théoricien du cloud.** Décrit la stack cible, pas la stack existante. *Question :* « Combien de services en production ne correspondent pas à ta description ? »
- **L'empileur de solutions.** A ajouté trois middlewares pour répondre à trois problèmes ponctuels. *Question :* « Combien de technologies distinctes pour le même usage ? »
- **Le minimaliste qui n'instrumente rien.** Produit du code sans observabilité. *Question :* « Combien de temps pour diagnostiquer un incident en production ? »
- **Le sécurité-par-feu.** Ajoute des couches de sécurité après coup, sans revoir l'architecture. *Question :* « Combien d'applications échouent à un test d'intrusion ? »

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les éditeurs (AWS vs Azure vs GCP). Rappeler que l'architecture technique est d'abord une question de principes, pas de produits.
- Ne pas confondre TRM et catalogue de produits. Le TRM est un modèle de référence ; le catalogue liste les produits utilisés.

Cas pratique — Architecture d'un jumeau numérique pour la maintenance prédictive (Aéronautique)

Contexte sectoriel : constructeur aéronautique européen, 12 000 employés, 45 lignes d'assemblage. La production est un actif critique : chaque ligne immobilisée représente une perte d'exploitation directe et un risque de retard de livraison aux compagnies aériennes clientes.

Diagnostic de l'existant. Sur les 45 lignes d'assemblage, **1 500 capteurs IoT** génèrent un flux massif de données (vibration, température, pression, acoustique) à une fréquence de 100 Hz. L'enjeu stratégique est de **réduire le taux de pannes sous les 3 %** pour éviter les **1,2 M€ de pertes d'exploitation annuelles** par chaîne arrêtée. Aujourd'hui, la maintenance est corrective ou planifiée à intervalles fixes, sans corrélation avec l'état réel des équipements.

Parties prenantes et écueil politique majeur.

Un point de friction existe entre deux populations :

- Le **Directeur Industriel** (focus ROI) — veut des résultats rapides, mesurables en pertes évitées. Il accepte la complexité technique tant qu'elle produit des gains financiers.
- L'équipe **IoT/Data** (focus complexité technique) — passionnée par les architectures élégantes (Kafka, time-series databases, machine learning), risque de privilégier la sophistication sur la valeur métier.

L'architecte doit **utiliser ArchiMate Motivation Extension** pour lier explicitement les Drivers (Coût des pannes) aux Goals (Maintenance Prédictive) aux Requirements (latence, fréquence d'échantillonnage). Cette traçabilité rend les arbitrages techniques audibles au Directeur Industriel.

Convergence IT/OT (Operational Technology).

L'architecture doit articuler deux couches historiquement disjointes :

- **Couche OT (Operational Technology)** : capteurs physiques, automates SCADA, bus de terrain industriels, contraintes temps réel strictes (latence < 10 ms).
- **Couche IT (Information Technology)** : Message Broker (Kafka), Cloud Data Lake, moteurs d'analyse, machine learning.

Le jumeau numérique est la **passerelle** entre les deux : il reçoit les événements OT, les historise, les modélise, et alimente les algorithmes de prédiction.

Décisions architecturales structurantes :

1. **Edge Computing** : les algorithmes de détection d'anomalie doivent tourner au plus près des capteurs (latence critique). On ne peut pas se permettre un aller-retour Cloud pour chaque échantillon à 100 Hz.
2. **Stockage time-series** : choisir une base optimisée (InfluxDB, TimescaleDB, OpenTSDB) plutôt qu'une base relationnelle classique. La volumétrie (1 500 capteurs × 100 Hz × 24h × 365j) exclut tout autre choix.
3. **Modèle de jumeau numérique** : chaque équipement critique est représenté par un agent logiciel qui agrège son état, son historique, et ses prédictions. Le modèle est exposé via API aux outils de maintenance.

Roadmap incrémentale (pas de Big Bang) :

- **Vague 1 (3 mois)** : instrumentation complète d'une ligne pilote, premiers modèles prédictifs simples (seuils statistiques).
- **Vague 2 (6 mois)** : déploiement sur 5 lignes, modèles de machine learning supervisés.
- **Vague 3 (12 mois)** : généralisation aux 45 lignes, enrichissement par maintenance prescriptive (recommandations d'action).

Livrables attendus de l'apprenant (schéma ArchiMate hybride) :

- Représentation sur trois couches : **Business** (capacité « Maintenance Prédictive »), **Application** (services d'analyse, jumeau numérique), **Technology** (capteurs, broker, base time-series, edge).
- Cartographie des flux temps réel (capteurs → broker → jumeau → algorithmes → tableaux de bord opérationnels).
- Cartographie des risques : obsolescence des automates, vulnérabilités cyber (un automate non patché devient un vecteur d'attaque majeur), latence réseau.

Pièges classiques à éviter :

- Oublier la **latence réseau** : sans Edge Computing, les modèles temps réel ne fonctionnent pas.
- **Vulnérabilité des automates industriels** non patchés face aux cybermenaces (rançongiciels ciblant l'OT en 2024-2025 : incidents documentés sur des sites manufacturiers européens).
- Chercher à tout faire en un projet : la roadmap doit être **incrémentale**, ligne par ligne, pour éviter tout arrêt de production.

Critères d'évaluation : le livrable est validé si l'apprenant (1) sépare explicitement couches IT et OT, (2) justifie le choix Edge Computing par une contrainte de latence chiffrée, (3) propose une roadmap qui n'arrête jamais la production.

Ce que la leçon 10 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- **Un TRM complet** documentant l'existant (sept catégories, criticité notée).
- **Un diagnostic** des composants à risque (obsolescence, dette, complexité).
- **Trois chantiers techniques** à 6 mois : abstraction, observabilité, sécurité by design.

Sources de cette leçon : TOGAF Series Guide to the Technical Reference Model (G229) · Charley Richard, *Mastering Cloud Migration* · Cas Mapify (anonymisé)

Leçon suivante : **L10 — Opportunités et Solutions (Phase E) : la cible priorisée.**

Quiz — 18 questions pour valider votre compréhension

Trois series de six questions. Prenez un crayon, écrivez vos reponses dans l'espace prevu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Qu'est-ce que le TRM (Technical Reference Model) selon l'Open Group ? Pourquoi décrit-il l'existant plutôt que la cible idéale ?

A2 Quels sont les trois niveaux d'abstraction de plate-forme (aucune, partielle, totale) ? En quoi le Niveau 2 ou 3 facilite-t-il la migration ?

A3 Pourquoi le middleware est-il un choix structurant (pas une commodité) ? Quelles sont les trois questions à poser avant tout choix de middleware ?

A4 Qu'est-ce que l'observabilité ? Quelles sont les trois questions auxquelles elle permet de répondre en production (métriques, logs, traces) ?

A5 Quels sont les cinq domaines de sécurité à traiter en architecture (identité, autorisation, chiffrement, audit, conformité) ?

A6 Pourquoi la sécurité doit-elle être conçue dès l'architecture (Security by Design) plutôt qu'ajoutée comme une couche après coup ?

B — AUTO-EVALUATION

B1 As-tu un TRM documenté pour ton organisation ? Décrit-il l'existant ou la cible ? Quelle est la dernière date de mise à jour ?

B2 Ton organisation est-elle au Niveau 1, 2 ou 3 d'abstraction ? Combien de temps faudrait-il pour migrer vers un autre cloud provider aujourd'hui ?

B3 Combien de middlewares coexistent dans ton organisation ? Sont-ils gouvernés par une équipe transverse ou par chaque application ? Combien de compétences différentes cela exige-t-il ?

B4 Combien de temps faut-il à ton équipe pour diagnostiquer un incident en production ? Combien d'outils d'observabilité coexistent ? As-tu des SLOs définis pour les services critiques ?

B5 Combien de revues d'architecture sécurité sont conduites chaque année dans ton organisation ? Combien d'applications sont concernées par chaque revue ?

B6 Décris en 5 minutes la stack technique de ton organisation. Combien de temps as-tu mis pour la documenter ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Produire un TRM documentant l'existant sur 7 catégories (applicatifs, données, infrastructure, plate-forme, sécurité, observabilité, déploiement) avec criticité. (Échéance : J+30.)

C2 Diagnostiquer les composants à risque (obsolescence, dette, complexité). (Échéance : J+45.)

C3 Lancer un chantier d'abstraction de plate-forme (Niveau 1 → 2 ou 3) sur 6 mois. (Échéance : J+180.)

C4 Instrumenter 3 services critiques avec OpenTelemetry, Grafana, et PagerDuty. (Échéance : J+60.)

C5 Définir 3 SLOs (Service Level Objectives) pour les services critiques et configurer les alertes. (Échéance : J+45.)

C6 Lancer un programme Security by Design : revue d'architecture sécurité systématique pour toute nouvelle application. (Échéance : J+60.)

Opportunités et Solutions (Phase E & F) : planifier la transition

Architecture d'Entreprise

IV — LE RAYONNEMENT

« A roadmap without priorities is a wish list. A roadmap without transition architectures is a fantasy. » —
The Open Group, TOGAF Standard — Phase E: Opportunities and Solutions
« La transition est un chemin, pas un bond. »

Vidéo d'introduction

 Télécharger cette leçon en PDF

Les dix projets qui ont failli ne jamais voir le jour

En 2014, Mutuelle Plus — la mutuelle dont nous avons parlé en L1 — entame la troisième année de sa transformation. Les deux premières années ont été consacrées à l'audit, à la cartographie, à la Phase Préliminaire, aux Phases A et B (Vision et Architecture Métier). Le diagnostic est posé, la cible est dessinée. Reste à planifier **comment** y aller.

C'est à ce moment qu'Élise, l'architecte d'entreprise, fait face à un piège classique : transformer une **cible architecturale** en une **liste de projets** qu'on peut confier aux équipes opérationnelles. Cette transformation n'est pas mécanique. Elle suppose de répondre à plusieurs questions structurantes :

- **Quels écarts sont les plus coûteux ?** L'analyse As-Is vs To-Be fait apparaître des dizaines d'écarts. Lesquels sont prioritaires ?
- **Quelles solutions couvrent plusieurs écarts ?** Un même projet peut répondre à trois écarts en même temps — ou trois projets peuvent répondre au même écart en se concurrençant.
- **Comment séquencer ?** Les projets ne sont pas indépendants. Certains sont des prérequis d'autres. Le séquençage est un problème d'optimisation, pas un problème de tri.
- **Comment gérer la transition ?** On ne peut pas basculer d'un coup d'un SI à un autre. Il faut des **architectures de transition** qui décrivent les états intermédiaires.

Élise passe trois mois à résoudre ces questions. Elle produit :

- Une **analyse d'écarts structurée** par couche (métier, données, applications, technique) avec une note de coût et de complexité pour chaque écart.
- Une **liste de Solution Building Blocks (SBB)** — des briques de solution qui couvrent les écarts, avec leur périmètre fonctionnel et leurs dépendances.
- **Sept architectures de transition** — des états intermédiaires qui maintiennent la cohérence du SI pendant la transformation.
- Une **roadmap 24 mois** avec trois vagues de chantiers, organisées par valeur métier et risques.

Ce travail a permis à Mutuelle Plus de passer d'une cible séduisante sur papier à un plan d'action exécutable — et de tenir la trajectoire pendant les six années suivantes.

C'est ce que cette leçon cherche à vous transmettre : **comment transformer une architecture cible en un plan de transition réaliste, sans céder à la tentation du grand soir.**

Le schéma suivant montre les sept architectures de transition qui ont permis à Mutuelle Plus de passer du SI legacy au SI cible sur 24 mois :

Premier principe — L'analyse d'écarts est As-Is vs To-Be, couche par couche

L'analyse d'écarts consiste à comparer l'architecture actuelle (As-Is) et l'architecture cible (To-Be) pour identifier les différences. Cette comparaison doit être faite couche par couche, pas comme une masse indistincte.

L'Open Group recommande une analyse d'écarts structurée en quatre couches :

1. **Couche métier** — quelles capacités sont absentes, sous-maturées, ou en sursurplus ?
2. **Couche données** — quelles données sont mal qualifiées, mal gouvernées, mal distribuées ?
3. **Couche applicative** — quelles applications manquent, sont obsolètes, ou font double emploi ?
4. **Couche technique** — quels composants techniques sont à risque, mal outillés, ou à renouveler ?

Chaque écart reçoit trois notes : **coût** (combien ça coûte à combler), **complexité** (combien c'est difficile techniquement), **valeur métier** (combien ça rapporte). Ces trois notes permettent de prioriser les écarts à traiter.

La contrefaçon classique consiste à produire une liste à plat de 200 écarts sans hiérarchie. Cette liste ne sert à rien — elle ne permet ni de décider, ni de budgéter, ni de planifier. La parade est de **classer les écarts** par criticité et de **regrouper les écarts similaires** en projets cohérents.

***Et toi ?** Combien d'écarts As-Is vs To-Be as-tu identifiés sur ton dernier projet ? Combien sont documentés avec une note de coût, complexité, et valeur ? Combien ont été transformés en projets ?*

Deuxième principe — Le Solution Building Block est la brique élémentaire

Le SBB (Solution Building Block) est une brique de solution qui couvre un ou plusieurs écarts. C'est l'unité élémentaire de la transition — plus fine que le projet, plus consistante que la tâche.

L'Open Group distingue deux types de briques :

- **ABB — Architecture Building Block** — la description **cible** d'un composant (métier, données, application, technique). C'est ce que l'architecture **prédit**.
- **SBB — Solution Building Block** — l'implémentation **choisie** pour réaliser un ABB. C'est ce que l'architecture **décide**.

Le SBB est l'unité qui sert à budgéter, à planifier, à contractualiser. Un SBB peut être :

- **Une application packagée** (un ERP, un CRM, un ETL)
- **Un développement spécifique** (un service, une API, un module)

- **Un composant SaaS** (un service cloud souscrit)
- **Une prestation** (du conseil, de la formation, de l'intégration)

La contrefaçon classique consiste à confondre SBB et projet. Un projet peut contenir plusieurs SBB ; un SBB peut s'étendre sur plusieurs projets. La parade est de **maintenir une bibliothèque de SBB** avec leur description, leur coût, leur délai, leur fournisseur, et leur état (à l'étude, choisi, en cours, livré, décommissionné).

***Et toi ?** Combien de SBB as-tu identifiés sur ton dernier projet ? Combien ont une bibliothèque tenue à jour ? Combien ont un fournisseur identifié ?*

Troisième principe — Les architectures de transition évitent les bonds

On ne passe jamais d'un coup d'un SI à un autre. Les architectures de transition décrivent les états intermédiaires qui maintiennent la cohérence pendant la transformation.

Une architecture de transition est un **état cible intermédiaire** du SI, qui coexiste avec des éléments anciens et des éléments nouveaux. Elle répond à trois questions :

- **Quelles applications anciennes restent actives pendant la transition ?** (et pour combien de temps)
- **Comment les nouvelles applications cohabitent-elles avec les anciennes ?** (interfaces, synchronisation, bascule progressive)
- **Quand et comment chaque composant ancien est-il décommissionné ?** (critères de bascule, plan de retrait)

La contrefaçon classique consiste à croire qu'on peut **basculer en un week-end** d'un système à un autre. Cette illusion est mortelle : elle sous-estime la complexité de la cohabitation, ignore les régressions, néglige la formation des utilisateurs. La parade est de **privilégier les bascules progressives** — un utilisateur, un service, une région à la fois — avec des architectures de transition explicites.

Pour Mutuelle Plus, Élise a défini **sept architectures de transition** sur 24 mois (voir schéma ci-dessus) :

- **T0** — état initial (le SI legacy)
- **T1** — MDM client en place (la donnée client est unifiée)
- **T2** — CRM en place (les processus front-office migrent)
- **T3** — ERP intégré (les processus back-office migrent)
- **T4** — BI en place (le reporting est unifié)
- **T5** — applications décommissionnées (la dette se résorbe)
- **T6** — état cible (le SI est stabilisé)

Chaque transition $T_n \rightarrow T_{n+1}$ a ses propres critères de bascule, ses propres tests de non-régression, ses propres KPIs de succès.

***Et toi ?** Combien d'architectures de transition as-tu définies sur ton dernier projet ? Combien sont documentées avec des critères de bascule explicites ? Combien de bascule ont été effectivement réalisées en une seule fois ?*

Quatrième principe — La roadmap est priorisée par valeur et risque

Une roadmap qui classe les chantiers par ordre chronologique sans logique de valeur et de risque est un calendrier, pas une stratégie.

L'Open Group recommande que la roadmap soit structurée en **trois vagues** :

- **Vague 1 — Quick wins** — chantiers à fort impact, faible complexité, court terme. Ils démontrent la valeur de l'architecture.
- **Vague 2 — Fondations** — chantiers structurants qui débloquent les chantiers suivants. Investissement lourd, valeur différée.
- **Vague 3 — Rayonnement** — chantiers d'optimisation, d'innovation, d'extension. Ils valorisent les fondations.

Chaque vague contient 5 à 10 chantiers. Chaque chantier a un **sponsor** identifié, un **indicateur de succès** chiffré, un **budget** estimé, et un **délai** cible.

La contrefaçon classique consiste à présenter la roadmap comme un Gantt de 24 mois sans hiérarchie. Cette vision plate ne permet pas de prendre des décisions de priorisation. La parade est de **maintenir la roadmap par valeur** — chaque chantier peut être reclassé si sa valeur se révèle plus ou moins importante que prévu.

Et toi ? Combien de vagues structurent ta roadmap ? Combien de quick wins ont été identifiés pour la Vague 1 ? Quel est le sponsor de chaque vague ?

Cinquième principe — Le Work Package est l'unité budgérable

Le Work Package (WP) est un lot de travaux qui peut être budgété, planifié et exécuté par une équipe. C'est l'unité qui relie l'architecture à l'opérationnel.

Un WP contient typiquement :

- **Un périmètre fonctionnel** — ce qui est livré
- **Des critères d'acceptation** — comment on sait que c'est terminé
- **Un budget** — estimé ou engagé
- **Un délai** — estimé ou engagé
- **Une équipe** — interne, externe, ou mixte
- **Des dépendances** — amont et aval dans la roadmap

La contrefaçon classique consiste à confondre WP et chantier, WP et SBB, WP et projet. La parade est de **maintenir un référentiel des WP** qui permet de suivre l'avancement, de détecter les dérives, et de communiquer avec le sponsor.

Pour Mutuelle Plus, Élise a tenu un **tableau de bord des WP** pendant 24 mois, avec un WP par chantier, une mise à jour hebdomadaire, et une revue mensuelle avec le COMEX. Ce suivi a permis de détecter tôt les dérives (chantier A en retard de 6 semaines → arbitrage sur le périmètre) et de célébrer les quick wins (chantier B livré en avance → communication positive).

Et toi ? Combien de WP as-tu définis sur ton dernier projet ? Combien ont un budget engagé ? Combien ont un sponsor identifié ? Quel est ton taux de livraison à l'heure ?

Application — Ma roadmap 90 jours

À produire séance tenante. Cinq sections précises.

1. **Analyse d'écarts** — 5 à 10 écarts prioritaires (As-Is vs To-Be), notés coût / complexité / valeur.
2. **SBB** — 2 à 3 Solution Building Blocks, chacun avec son périmètre et son fournisseur envisagé.
3. **Architectures de transition** — 2 à 3 états intermédiaires, avec critères de bascule.
4. **Roadmap** — 3 vagues (quick wins / fondations / rayonnement), 5 à 10 chantiers par vague.
5. **Work Packages** — 1 WP par chantier prioritaire, avec budget, délai, sponsor, équipe.

Cette roadmap est ton premier livrable de Phases E-F. Elle rend l'architecture actionnable.

Module complémentaire — Le langage financier de l'architecte : TCO, Business Case et pitch COMEX

Une roadmap de transition ne vaut rien si elle n'est pas traduite dans le vocabulaire du décideur. L'architecte d'entreprise qui présente un schéma de capacités à un COMEX passe pour un technicien ; celui qui présente un **TCO sur cinq ans, un Coût de l'inaction chiffré et trois options comparées** passe pour un partenaire stratégique. Ce module donne à l'architecte les outils financiers nécessaires pour transformer une trajectoire technique en décision exécutive.

La structure du TCO sur cinq ans

Le **TCO** (*Total Cost of Ownership*) remplace la vision « coût projet » par une vision de possession sur cinq ans. Pour Mutuelle Plus, dont le chiffre d'affaires plafonne à 100 M€ face à des géants pesant des milliards, chaque décision d'architecture doit être un arbitrage sur la préservation des fonds propres (1 Md€) et le maintien de la marge de solvabilité.

Décomposition CAPEX / OPEX :

Composante	Nature	Éléments inclus pour Mutuelle Plus
CAPEX (Investissement)	Immobilisations	Licences, infrastructures Cloud (IaaS), coûts de conception (ADM Phases A-D), migration des données clients
OPEX (Exploitation)	Charges courantes	Abonnements SaaS, maintenance corrective et évolutive (Phase H), support N1/N2/N3, formation des 800 collaborateurs

Coûts cachés à quantifier explicitement :

- **Vendor Lock-in subi** : la rigidité de Gest+ empêche toute négociation tarifaire et bloque l'agilité métier. C'est un coût invisible mais réel.
- **Passifs de non-conformité** : le retard sur le RGPD constitue une provision pour risque juridique et financier au bilan.
- **Dette d'intégration** : le coût de maintenance du Shadow IT (tableurs par agence) représente un surcoût opérationnel direct.

Formule du TCO sur 5 ans :

$$TCO_{5\text{ ans}} = \Sigma CAPEX_{\text{initiaux}} + \Sigma (OPEX_{\text{annuels}} / (1+i)^n)$$

où i représente le taux d'actualisation financier. L'actualisation reflète la valeur temps de l'argent : un euro dépensé dans trois ans coûte moins qu'un euro dépensé aujourd'hui.

Le Coût de l'Inaction (COI) — la traduction financière de la dette technique

Le DAF ne raisonne pas en « bugs » mais en « érosion de la marge de solvabilité ». L'architecte doit traduire le ressenti technique en **Coût de l'Inaction** (COI), c'est-à-dire ce que l'organisation perd chaque année à ne pas transformer.

Exemple chiffré — Mutuelle Plus :

Source d'inefficacité	Calcul	Total annuel
Coût de la non-qualité (incidents)	500 erreurs de remboursement × 200 €/incident	100 000 €/an
Coût du ralentissement (productivité)	10 000 tickets × 5h supplémentaires × 50 €/h	250 000 €/an
Perte de vitesse portefeuille (résiliations faute de réactivité)	1 % d'adhérents perdus × ~300 € de cotisation moyenne	3 000 000 €/an
Total COI estimé		~3 350 000 €/an

Pour le COMEX, ce chiffre n'est pas une dépense informatique mais une **perte sèche de 3 % du chiffre d'affaires annuel**, menaçant directement les fonds propres. C'est cette traduction qui fait basculer le débat du « luxe qu'on ne peut pas se permettre » vers l'« urgence économique à traiter ».

Le template de Business Case simplifié (une page)

Le Business Case est le support d'arbitrage ultime pour débloquer les budgets de transformation. Il tient en une page et présente trois options chiffrées.

Structure type :

Section	Contenu
Contexte stratégique	Nécessité de restaurer la « solidarité intergénérationnelle » en attirant des cotisants jeunes. Alignement du SI pour servir les agents territoriaux.
Options étudiées	1. <i>Do Minimum</i> — maintenance Gest+ (COI ~3 M€/an). 2. <i>Cible intermédiaire</i> — réhabilitation partielle. 3. <i>Cible idéale</i> — CRM SaaS moderne.
KPIs financiers	ROI sur 3 ans, période de payback, TCO à 5 ans.
Risques résiduels	Adoption utilisateur (l'âge moyen de 43 ans nécessite un plan de formation massif).

Le pitch COMEX en trois slides (dix minutes)

L'objectif est d'obtenir un arbitrage en dix minutes en parlant « valeur métier ».

Slide 1 — L'effet ciseau financier. *Message* : Notre SI est le goulot d'étranglement qui nous empêche de créer des produits spécifiques. *Visuel* : graphique en « ciseau » montrant la hausse des coûts de couverture (vieillessement)

croisant la baisse des cotisations (perte de jeunes adhérents).

Slide 2 — Trajectoires de valeur. *Message* : L’inaction nous coûte 15 M€ sur 5 ans. L’investissement de 2,5 M€ est amorti en 14 mois. *Visuel* : comparaison des courbes de TCO (Statu Quo vs CRM SaaS).

Slide 3 — Recommandation de l’architecte. *Message* : Lancer le projet CRM pour protéger notre milliard d’euros de fonds propres et automatiser la gestion des réclamations. *Visuel* : tableau de bord prospectif liant la technologie à la marge de solvabilité.

Trois pièges fatals à éviter :

- 1. **Le jargon technique.** Évitez « Micro-services », dites « Capacité d’évolution modulaire ».
- 2. **L’absence de chiffres.** Ne parlez jamais de « besoin », parlez de « perte sèche ».
- 3. **Le manque d’alternative.** Quantifiez toujours l’option « Inaction » pour souligner le danger.

Étude de cas chiffrée — Refonte du CRM de Mutuelle Plus

Mutuelle Plus (800 collaborateurs, 1 million d’assurés) souffre d’une fragmentation de la donnée client, chaque agence gérant ses propres tableurs.

Critère	Statu Quo (Gest+)	Réhabilitation partielle	CRM SaaS moderne
Investissement (CAPEX)	0 €	1,2 M€	2,5 M€
Dette technique (COI/an)	3 M€	1,5 M€	0,2 M€
TCO sur 5 ans	15 M€	8,7 M€	3,5 M€
Payback Period	N/A	30 mois	14 mois
ROI sur 3 ans	Négatif	~25 %	~45 %

Lecture : l’option CRM SaaS divise le TCO par quatre par rapport au statu quo. C’est le levier indispensable pour attirer les cotisants jeunes et stabiliser la base client. L’architecte dispose ici d’un argument financier béton pour défendre la cible.

Le « Shift Left Finance » — intégrer la gouvernance FinOps dans le cycle ADM

Pour garantir la pérennité financière, l’AE pratique le *Shift Left Finance* : intégrer le contrôle des coûts dès la conception.

Cartographie ADM ↔ FinOps :

- **Phase C (SI)** : distinction entre ABB et SBB. Sélection des briques de solution selon leur efficience énergétique et financière.
- **Phase E (Opportunités)** : arbitrage financier. Choix des solutions Cloud basé sur la consommation réelle plutôt que sur le sur-provisionnement historique.
- **Phase F (Planning)** : utilisation de l’Architecture de Transition. On définit des paliers incrémentaux qui délivrent une valeur métier immédiate (ex. : numérisation des contrats d’abord) pour autofinancer la suite de la transformation.
- **Phase G (Gouvernance)** : boucle FinOps (Inform → Optimize → Operate). Suivi en temps réel de la dérive budgétaire.

Cette gouvernance garantit que l’architecture reste un **moteur de rentabilité**, pas un poste de coût.

Synthèse

Maîtriser les chiffres n'est pas optionnel pour l'architecte d'entreprise. C'est ce qui transforme un schéma directeur en un dossier d'investissement, et un dossier d'investissement en une décision exécutive. Le COMEX ne raisonne pas en « conformité aux principes TOGAF » ; il raisonne en **marge de solvabilité, ROI, payback, risque de non-conformité**. L'architecte qui parle ce langage devient un partenaire stratégique ; celui qui ne le parle pas reste un prestataire technique.

Encadré Au lecteur / À l'animateur

Question d'ouverture : « *Combien d'écarts As-Is vs To-Be as-tu identifiés sur ton dernier projet ?* »

Profils à observer :

- **Le sauteur.** Voudrait basculer en un week-end d'un SI à un autre. *Question : « Combien d'architectures de transition as-tu définies ? »*
- **Le collectionneur d'écarts.** Produit une liste de 200 écarts sans hiérarchie. *Question : « Combien d'écarts sont notés coût / complexité / valeur ? »*
- **Le projet par chantier.** Confond WP et SBB, WP et projet. *Question : « Combien de WP ont un budget engagé ? »*
- **Le planneur plat.** Présente la roadmap comme un Gantt sans hiérarchie. *Question : « Combien de vagues structurent ta roadmap ? »*

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les outils (MS Project, Jira, etc.). Rappeler que la roadmap est d'abord un document stratégique.
- Ne pas confondre architecture de transition et architecture cible. La transition décrit un état intermédiaire, la cible décrit l'état final.

Ce que la leçon 11 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- **Une analyse d'écarts** structurée par couche, avec 5 à 10 écarts prioritaires.
- **Une bibliothèque de 2 à 3 SBB** avec leurs caractéristiques.
- **Une roadmap 3 vagues** avec 5 à 10 chantiers par vague.

Sources de cette leçon : TOGAF Standard 10^e (Phases E-F) · TOGAF Series Guide to Business Capability Planning (G233e)
· Cas Mutuelle Plus (anonymisé)

Leçon suivante : **L11 — Gouvernance de l'Implémentation (Phase G) : assurer la conformité.**

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Qu'est-ce que l'analyse d'écarts As-Is vs To-Be ? Pourquoi doit-elle être structurée par couche (métier / données / applicatif / technique) ?

A2 Quelle est la différence entre un ABB (Architecture Building Block) et un SBB (Solution Building Block) ? Quel est le rôle du SBB dans la transition ?

A3 Qu'est-ce qu'une architecture de transition ? Pourquoi faut-il en définir plusieurs, plutôt que de basculer en un seul grand soir ?

A4 Quels sont les trois niveaux de vagues (quick wins / fondations / rayonnement) ? Comment s'articulent-ils ?

A5 Qu'est-ce qu'un Work Package (WP) ? Quels sont ses six éléments constitutifs ?

A6 Qu'est-ce que la priorisation par valeur et risque ? Pourquoi est-elle préférable à une priorisation chronologique ?

B — AUTO-EVALUATION

B1 Combien d'écarts As-Is vs To-Be as-tu identifiés sur ton dernier projet ? Combien sont documentés avec une note de coût, complexité, et valeur ? Combien ont été transformés en projets ?

B2 Combien de SBB as-tu identifiés sur ton dernier projet ? Combien ont une bibliothèque tenue à jour ? Combien ont un fournisseur identifié ?

B3 Combien d'architectures de transition as-tu définies sur ton dernier projet ? Combien sont documentées avec des critères de bascule explicites ?

B4 Combien de vagues structurent ta roadmap ? Combien de quick wins ont été identifiés pour la Vague 1 ? Quel est le sponsor de chaque vague ?

B5 Combien de WP as-tu définis sur ton dernier projet ? Combien ont un budget engagé ? Combien ont un sponsor identifié ? Quel est ton taux de livraison à l'heure ?

B6 Sur ton dernier projet de transition, combien d'états intermédiaires as-tu définis ? Combien sont documentés avec des critères de bascule explicites ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Produire une analyse d'écarts As-Is vs To-Be structurée par couche, avec 5 à 10 écarts prioritaires notés coût / complexité / valeur. (Échéance : J+14.)

C2 Identifier 2 à 3 Solution Building Blocks, chacun avec son périmètre et son fournisseur envisagé. (Échéance : J+21.)

C3 Définir 2 à 3 architectures de transition, avec critères de bascule documentés. (Échéance : J+30.)

C4 Construire une roadmap 3 vagues avec 5 à 10 chantiers par vague, sponsor identifié pour chaque vague. (Échéance : J+45.)

C5 Rédiger un WP par chantier prioritaire : périmètre, critères d'acceptation, budget, délai, équipe, dépendances. (Échéance : J+45.)

C6 Tenir un tableau de bord des WP avec mise à jour hebdomadaire et revue mensuelle avec le sponsor. (Échéance : J+60.)

Gouvernance de l'Implémentation (Phase G) : assurer la conformité

Architecture d'Entreprise

IV — LE RAYONNEMENT

« Architecture governance is not about saying no. It is about ensuring that every architectural decision is informed, traceable, and reversible. » — The Open Group, TOGAF Standard — Phase G: Implementation Governance

« Un architecte qui refuse tout n'est pas un architecte. C'est un censeur. »

Vidéo d'introduction

📄 Télécharger cette leçon en PDF

La gouvernance qui a sauvé Mutuelle Plus — et celle qui l'aurait tuée

En 2015, Mutuelle Plus — toujours elle — entre dans la Phase G, la gouvernance de l'implémentation. Élise a passé quatre ans à produire l'architecture cible, à planifier la transition, à négocier les chantiers. Les équipes projet ont commencé à exécuter. Et là, **un nouveau piège** apparaît : chaque équipe projet interprète l'architecture à sa manière, certaines décisions sont contournées, et la cible se met à dériver.

Élise découvre trois symptômes typiques d'une gouvernance défailante :

- **Les dispenses** — les projets sollicitent des dispenses pour ne pas respecter l'architecture. Sans processus clair, les dispenses s'accumulent, deviennent un précédent, et vident l'architecture de sa substance.
- **Les revues de conformité** — certains projets affirment qu'ils sont conformes sans avoir été testés formellement. La conformité par affirmation est une fiction.
- **Les contrats d'architecture** — les projets ne savent pas ce qu'ils doivent livrer comme livrables architecturaux. Les architectes passent leur temps à demander des compléments.

Élise a l'intelligence de ne pas répondre par la répression. Elle instaure trois mécanismes :

1. **Un processus de dispense explicite** — toute dispense doit être demandée, justifiée, et tracée. Pas de dispense par défaut.
2. **Des revues de conformité formelles** — chaque projet passe par une revue en trois points : (a) revue de conception, (b) revue d'implémentation, (c) revue de mise en production.
3. **Des contrats d'architecture signés** — chaque projet signe avec l'architecture un contrat qui définit les engagements réciproques (ce que l'architecture fournit, ce que le projet s'engage à respecter).

Ce qui aurait pu être un casse-tête bureaucratique est devenu un **cadre de confiance** : les projets savent ce qu'on attend d'eux, l'architecture sait ce qu'elle doit fournir, et les litiges sont tranchés par les contrats signés.

C'est ce que cette leçon cherche à vous transmettre : **comment la gouvernance de l'implémentation protège l'architecture sans paralyser les projets, en posant un cadre clair et en l'appliquant avec discernement.**

Le schéma suivant résume les trois moments où l'architecture intervient dans un projet, et le processus en quatre étapes pour gérer les dispenses :

Premier principe — L'architecture est un garde-fou, pas une police

Le rôle de l'architecture en Phase G n'est pas de bloquer les projets. C'est de poser un cadre qui permet aux projets d'avancer vite, sans mettre en péril la cohérence du SI.

La tentation du « garde-fou » est de tout refuser. Cette posture rend l'architecture infréquentable et les projets hostiles. La parade est d'**accompagner les projets** plutôt que de les bloquer. L'architecte devient un **conseiller** dont l'avis est sollicité en amont, pas un **censeur** dont l'avis est requis en aval.

L'Open Group recommande que la gouvernance s'exerce à **trois moments** :

1. **Revue de conception** — l'architecte valide la conception avant le démarrage du développement. C'est le moment le plus efficace : corriger un schéma est moins coûteux que corriger du code.
2. **Revue d'implémentation** — l'architecte vérifie que l'implémentation respecte la conception. C'est le moment où les écarts apparaissent.
3. **Revue de mise en production** — l'architecte valide le déploiement en production. C'est le moment où la dette technique se crée si on n'a pas fait les deux premières revues.

La contrefaçon classique consiste à ne faire que la revue de mise en production — c'est-à-dire quand il est trop tard. La parade est de **planifier les trois revues** dans le calendrier du projet, dès la phase de cadrage.

***Et toi ?** Combien de revues d'architecture sont planifiées sur chacun de tes projets ? À quels moments du cycle de vie ? Combien sont effectivement menées à terme ?*

Deuxième principe — Le contrat d'architecture est un engagement récipro

Le contrat d'architecture n'est pas un document que l'architecture impose aux projets. C'est un accord entre l'architecture et le projet, qui engage les deux parties.

Un contrat d'architecture documente quatre engagements :

- **Ce que l'architecture fournit au projet** — le périmètre architectural, les exigences non fonctionnelles, les interfaces avec les autres systèmes, les contraintes techniques.
- **Ce que le projet s'engage à respecter** — la conformité aux exigences, les patterns techniques imposés, les contrats d'interface à respecter.
- **Comment les conflits sont tranchés** — l'escalade vers le comité d'architecture, les dispenses, les dérogations.
- **Quels livrables le projet doit produire** — les modèles ArchiMate, les contrats d'interface, les tests de non-régression.

La contrefaçon classique consiste à produire un contrat d'architecture que personne ne lit, ou que le projet signe sans l'avoir compris. La parade est de **limiter le contrat à 2 pages**, de le **signer conjointement** (architecte ET chef de

projet), et de le **relire à chaque revue**.

Pour Mutuelle Plus, Élise a tenu un **catalogue de contrats signés** — un fichier partagé listant tous les projets en cours, avec leur contrat, leur état (signé / en cours / terminé), et leurs dispenses éventuelles. Ce catalogue permettait à tout acteur de savoir où en était chaque projet vis-à-vis de l'architecture.

***Et toi ?** Combien de projets en cours ont un contrat d'architecture signé ? Combien ont une dispense active ? Combien ont une revue planifiée ?*

Troisième principe — La revue de conformité est formelle, pas par affirmation

La conformité « parce que le chef de projet l'a dit » n'est pas une conformité. La conformité est un état vérifié, pas un état supposé.

L'Open Group recommande que chaque revue de conformité suive un processus en trois étapes :

1. **Préparation** — le projet fournit les livrables à l'architecture (schémas, contrats d'interface, tests). L'architecture prépare la grille de revue.
2. **Confrontation** — l'architecte passe en revue les livrables, pose des questions, identifie les écarts. Le projet répond.
3. **Décision** — l'architecte émet un avis (conforme / conforme avec réserves / non conforme). L'avis est tracé, daté, signé.

La contrefaçon classique consiste à faire des revues de conformité par e-mail, sans grille de revue, sans confrontation. Ces revues sont des formalités administratives. La parade est de **tenir les revues en présentiel** (ou en visio), avec un **déroulé strict**, et de **documenter les écarts**.

Pour Mutuelle Plus, Élise a utilisé un **template de revue** d'une page, avec trois colonnes (livrable attendu, livrable reçu, écart constaté). Le template est rempli pendant la revue, signé par les deux parties, et archivé dans le catalogue des revues.

***Et toi ?** Combien de revues de conformité as-tu menées sur les 12 derniers mois ? Combien ont produit un avis tracé ? Combien ont identifié des écarts significatifs ?*

Quatrième principe — Les dispenses sont gérées, pas interdites

Une dispense n'est pas un échec de l'architecture. C'est un acte de gestion, qui reconnaît un écart entre la cible et la réalité, et qui le trace.

L'Open Group recommande un **processus de dispense** explicite, en quatre étapes :

1. **Demande** — le projet soumet une demande de dispense, avec justification (pourquoi l'écart, quel impact, quelle durée).
2. **Évaluation** — l'architecture évalue la demande (impact sur la cible, risques techniques, alternatives).
3. **Décision** — le comité d'architecture accepte, refuse, ou accepte avec conditions.
4. **Suivi** — la dispense est tracée, et le projet doit revenir vers l'architecture quand la condition est levée.

La contrefaçon classique consiste à interdire les dispenses, ce qui pousse les projets à des contournements officieux. La parade est d'**accepter un certain volume de dispenses**, à condition qu'elles soient tracées et résolues dans un délai.

Pour Mutuelle Plus, Élise a tenu un **registre des dispenses** — un fichier listant toutes les dispenses actives, avec leur statut (en attente / acceptée / résolue / expirée). Le registre est revu tous les mois au comité d'architecture. Les dispenses non résolues après 6 mois déclenchent une revue prioritaire.

***Et toi ?** Combien de dispenses actives dans ton organisation ? Combien ont été résolues dans les 6 mois ? Combien sont restées ouvertes plus de 12 mois ?*

Cinquième principe — La Phase H gère le changement continu

La Phase H (Architecture Change Management) ne signifie pas que l'architecture est terminée. Elle signifie que l'architecture entre dans un cycle de gestion continue du changement.

L'Open Group recommande que la Phase H s'organise autour de cinq activités :

- **Surveillance** — observer les changements dans l'environnement (technologie, réglementation, marché).
- **Évaluation** — évaluer l'impact des changements sur l'architecture.
- **Recommandation** — recommander des mises à jour de l'architecture si nécessaire.
- **Implémentation** — coordonner la mise en œuvre des changements architecturaux.
- **Communication** — tenir les parties prenantes informées des changements.

La contrefaçon classique consiste à considérer l'architecture comme un livrable terminé. Une fois la cible définie, l'architecture est « fermée » et personne ne la maintient. La parade est de **maintenir l'architecture comme un objet vivant** — révisé annuellement, mis à jour en continu, et consulté pour toute décision structurante.

Pour Mutuelle Plus, Élise a institué une **revue annuelle de l'architecture** — trois jours, avec les propriétaires de domaine, les sponsors, et quelques experts externes. Cette revue a permis de réajuster la cible à trois reprises sur six ans — et de maintenir la cohérence de la transformation.

***Et toi ?** Combien de revues annuelles de l'architecture sont tenues dans ton organisation ? Combien de mises à jour de la cible sont décidées par an ? Combien de demandes de changement sont arbitrées par an ?*

Application — Mon contrat d'architecture

À produire séance tenante. Un contrat d'architecture d'**une page** pour un projet réel (en cours ou à venir).

1. **Périmètre** — nom du projet, sponsor, chef de projet, dates clés.
2. **Engagements de l'architecture** — ce que l'architecture fournit (schémas, exigences, contrats d'interface).
3. **Engagements du projet** — ce que le projet respecte (patterns, normes, contrats).
4. **Dispenses** — liste des dispenses éventuelles, avec conditions.
5. **Revue planifiée** — dates des trois revues (conception, implémentation, production).
6. **Escalade** — qui tranche en cas de désaccord.

Ce contrat est ton premier livrable de Phase G. Il pose le cadre de la collaboration entre l'architecture et le projet.

Module complémentaire — L’architecte d’entreprise dans un monde agile : TOGAF × SAFe

Les organisations qui réussissent leur transformation numérique combinent deux logiques longtemps présentées comme opposées : la **rigueur stratégique** de TOGAF (vision long terme, cohérence globale, cycle ADM) et la **vélocité d’exécution** des méthodes agiles à l’échelle (SAFe, Spotify, LeSS). L’architecte d’entreprise du XXIe siècle n’a pas le choix : il doit **orchestrer** les deux. Ce module présente l’articulation opérationnelle entre ADM et SAFe, à travers le prisme de Mutuelle Plus confrontée à la coexistence entre son cycle de transformation et son train agile.

Mapping ADM ↔ SAFe — l’alignement stratégique

Pour qu’un cycle ADM et un train agile (ART) collaborent réellement, chaque phase ADM doit produire des artefacts consommables par le backlog agile, et inversement. Le tableau ci-dessous décrit ce flux d’informations critique :

Phase TOGAF ADM	Artefact SAFe consommé / produit	Application concrète
Phase A (Vision)	Portfolio Vision, Epic Hypothesis	L’AE transforme les objectifs de reconquête de Mutuelle Plus en Epics : passer de 72h à l’immédiateté, reconquérir les jeunes cotisants.
Phase B (Métier)	Development Value Streams	Modélisation des processus pivots (cotisations, prestations, contrats). Identification des étapes manuelles (recherches de contrats non numérisés) pour alimenter les Business Enablers.
Phases C & D (SI & Technique)	Architecture Runway	Les modèles logiques de données ciblent la résolution des doublons dans Gest+. Les architectures applicatives définissent les NFR (performance, sécurité RGPD).
Phases E & F (Transition)	PI Roadmap	Corrélation des paliers de transition (socle d’identité avant Espace Client) avec la planification des Program Increments.

Cet alignement constitue la structure porteuse de la **piste d’atterrissage** technique. Sans elle, l’agilité se transforme en succession de rustines tactiques sur un SI déjà rigide.

Architecture Runway vs Enabler — deux concepts distincts

L’**Architecture Runway** est l’état physique existant du socle technique : APIs, plateformes Cloud, intégrité des données, briques d’authentification. C’est ce qui permet aux équipes agiles de développer à pleine vitesse, sans être freinées par un mur technique.

L’**Enabler** est l’unité de travail dans le backlog (Epic, Feature ou Story) destinée à **construire ou maintenir** cette piste. L’Enabler est ce que l’AE inscrit dans le backlog pour garantir que la Runway avance au rythme des besoins métiers.

Découpage industriel des Runways pour Mutuelle Plus :

- **Socle de données unifié (Projet #2)** — Enabler visant l’élimination des doublons Gest+ par un MDM (Master Data Management) pour restaurer la fiabilité des remboursements.
- **Hub CRM Transverse (Projet #6)** — Mise en place d’une architecture permettant la mobilité géographique des

adhérents entre agences, supprimant la dépendance aux fichiers Excel locaux.

- **IAM (Identity & Access Management) — Projets #3 & #10** — Architecture microservices pour l'Espace Client sécurisé, priorisant la simplicité d'authentification pour les assurés et les gestionnaires.

Ces Runways sont les **livrables architecturaux** que l'AE porte dans les PI Plannings successifs.

Le rôle de l'AE pendant le PI Planning

Le PI Planning est le moment où l'architecture « tour d'ivoire » se confronte à la réalité des équipes. L'AE y intervient en tant que **serviteur-leader**, garant de la cohérence globale.

Préparation (en amont du PI Planning) :

1. Rédaction des critères de succès techniques — définition stricte des NFR (ex. : scalabilité pour supporter 1 million d'adhérents).
2. Identification des dépendances — cartographie des liens entre le train « Front-End » et le train « Core Insurance » pour éviter les blocages sur les flux Gest+.
3. Estimation du **WSJF** (*Weighted Shortest Job First*) pour prioriser les Enablers. L'AE doit utiliser la donnée financière pour peser face aux demandes métier : pour Mutuelle Plus (100M€ de CA), le **Coût du Délai** d'un Enabler se chiffre.
Exemple : retarder l'Enabler « CRM Transverse » empêche le traçage des résiliations. Si ce manque de visibilité cause une perte de 1 % des adhérents par an, le Coût du Délai est de **1 million d'euros par an**. Ce chiffre permet de prioriser l'Enabler face à une Feature métier mineure.

Facilitation (Jour 1 et 2 du PI Planning) :

- **Éclairer** : orienter les choix techniques pour éviter le développement de solutions « ad hoc » qui aggraveraient la dette technique.
- **Arbitrer** : trancher les impasses immédiates (ex. : choix d'un protocole d'échange).
- **Valider les compromis** : accepter une solution tactique pour tenir l'objectif du PI, à condition qu'elle soit documentée et que sa résolution soit planifiée dans la trajectoire TOGAF.

L'AE ne **commande** pas l'équipe ; il **alimente** ses décisions avec les contraintes architecturales globales. C'est ce qui le distingue du chef de projet.

System Demo, Inspect & Adapt et Contrat d'Architecture

La gouvernance agile chez Mutuelle Plus doit s'assurer que l'autonomie des équipes ne dégrade pas la solvabilité ou la conformité RGPD.

System Demo et Architecture Contract. Lors de la démonstration de fin de sprint, l'AE valide la conformité des réalisations par rapport au **Contrat d'Architecture** (issu de la Phase G de TOGAF). C'est ici que l'on vérifie physiquement que les NFR sont respectées et que la Runway est opérationnelle.

Inspect & Adapt (I&A). L'AE analyse les « irritants systémiques » (ex. : lenteurs récurrentes de Gest+ impactant le Centre de Services). Ces problèmes sont transformés en nouveaux Enablers pour le PI suivant. C'est la **boucle d'apprentissage** qui garantit que la trajectoire TOGAF absorbe le retour du terrain.

Étude de cas — Trajectoire sur trois Program Increments

Illustration de la transition vers un système d'identité client moderne chez Mutuelle Plus :

PI 1 — Les fondations. Focus sur l'Enabler d'infrastructure (OAuth2/OIDC). - *Livrables TOGAF* : Architecture Definition Document (partiel), Principes d'Architecture mis à jour pour inclure la sécurité « Zero Trust ».

PI 2 — La première valeur métier. Livraison de la Feature « Espace Client sécurisé » exploitant le socle du PI 1. - *Livrable TOGAF* : Architecture Transition v1 (état réel après migration partielle).

PI 3 — L'accélération industrielle. Livraison de « Paiement en un clic » et « Profil enrichi » sans re-conception. -

Livrable TOGAF : Architecture Contract. L'AE utilise ce document pour signer la conformité des Features lors de la System Demo, clôturant ainsi le cycle de gouvernance.

Trois écueils de cohabitation AE / Agile

La transformation risque de trébucher sur trois dérives classiques :

Le goulot d'étranglement (Ivory Tower). L'AE ralentit les équipes par un excès de validation. *Remède* : définir des NFR claires et laisser l'autonomie de réalisation tant que les gardes-fous sont respectés. Le contrat d'architecture prime sur la présence quotidienne.

L'AE dilué ou disparu. L'absence de vision de cible génère une dette technique anarchique, mettant en péril la fiabilité des remboursements et la solvabilité à long terme. *Remède* : sanctuariser le rôle d'AE dans la gouvernance, même quand l'agilité semble tout résoudre.

La guerre AE vs Product Owner. Conflit entre vision long terme et livraison immédiate. *Remède* : instaurer des ratios budgétaires sanctuarisés. Pour Mutuelle Plus, dédier d'office **20 % de la capacité de chaque train (ART) aux Enablers d'architecture**. Cela garantit l'entretien de la Runway et évite l'obsolescence du SI.

Synthèse

L'architecte moderne n'est plus un censeur. C'est un **serviteur-leader** : il est le garant de la pérennité du patrimoine de l'entreprise, en s'assurant que l'agilité d'aujourd'hui ne devienne pas la rigidité de demain. Le couple TOGAF x SAFe n'est pas un compromis ; c'est une **complémentarité structurelle** entre la stratégie de long terme et l'exécution de court terme.

Encadré Au lecteur / À l'animateur

Question d'ouverture : « *Combien de projets en cours dans ton organisation ont un contrat d'architecture signé ?* »

Profil à observer :

- **Le censeur.** Refuse tout, bloque tous les projets. *Question* : « *Combien de dispenses as-tu acceptées cette année ?* »
- **Le laxiste.** Ne fait pas de revue, accepte tout. *Question* : « *Combien de revues as-tu menées à terme ?* »
- **Le procédurier.** Fait des revues formelles sans fond. *Question* : « *Combien d'écarts significatifs as-tu identifiés cette année ?* »
- **L'isolé.** N'a personne avec qui signer les contrats. *Question* : « *Combien de chefs de projet ont signé un contrat avec toi ?* »

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les outils de ticketing (Jira, ServiceNow). Rappeler que la gouvernance est d'abord une pratique, pas un outil.
- Ne pas confondre gouvernance et audit. La gouvernance est continue ; l'audit est ponctuel.

Ce que la leçon 12 vous demande

Ouvrez votre dossier d'architecte à la page suivante. Préparez :

- **Un contrat d'architecture** signé pour un projet en cours.
- **Un registre des dispenses** avec leur statut.

- **Un calendrier des revues** pour les 3 mois à venir.

Sources de cette leçon : TOGAF Standard 10^e (Phases G-H) · COBIT 2019 · ISACA, *COBIT 2019 Framework* · Cas Mutuelle Plus (anonymisé)

Leçon suivante : **L12 — Durabilité numérique et éco-conception.**

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Pourquoi l'architecture en Phase G doit-elle être un garde-fou plutôt qu'une police ? Comment cette posture change-t-elle la relation avec les projets ?

A2 Qu'est-ce qu'un contrat d'architecture ? Quels sont les quatre engagements qu'il documente ? Pourquoi doit-il être limité à 2 pages ?

A3 Pourquoi la revue de conformité doit-elle être formelle (pas par affirmation) ? Quelles sont les trois étapes d'une revue ?

A4 Qu'est-ce qu'une dispense d'architecture ? Pourquoi faut-il la gérer plutôt que l'interdire ? Quelles sont les quatre étapes du processus ?

A5 Qu'est-ce que la Phase H (Architecture Change Management) ? Quelles sont les cinq activités qui la composent ?

A6 Quels sont les trois moments où l'architecture intervient dans un projet (revue de conception / d'implémentation / de mise en production) ?

B — AUTO-EVALUATION

B1 Combien de revues d'architecture sont planifiées sur chacun de tes projets ? À quels moments du cycle de vie ? Combien sont effectivement menées à terme ?

B2 Combien de projets en cours ont un contrat d'architecture signé ? Combien ont une dispense active ? Combien ont une revue planifiée ?

B3 Combien de revues de conformité as-tu menées sur les 12 derniers mois ? Combien ont produit un avis tracé ? Combien ont identifié des écarts significatifs ?

B4 Combien de dispenses actives dans ton organisation ? Combien ont été résolues dans les 6 mois ? Combien sont restées ouvertes plus de 12 mois ?

B5 Combien de revues annuelles de l'architecture sont tenues dans ton organisation ? Combien de mises à jour de la cible sont décidées par an ?

B6 Combien de projets en cours dans ton organisation ont un contrat d'architecture signé ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Rédiger un contrat d'architecture d'1 page pour un projet en cours, signé conjointement. (Échéance : J+14.)

C2 Planifier les 3 revues d'architecture (conception / implémentation / production) dans le calendrier du projet. (Échéance : J+30.)

C3 Tenir un registre des dispenses avec leur statut (en attente / acceptée / résolue / expirée), revue mensuelle. (Échéance : J+21.)

C4 Créer un template de revue de conformité en 1 page, avec colonnes (livrable attendu / reçu / écart). (Échéance : J+30.)

C5 Tenir un catalogue de contrats d'architecture signés, accessible à tous les acteurs. (Échéance : J+45.)

C6 Programmer une revue annuelle de l'architecture avec les propriétaires de domaine et les sponsors.
(Échéance : J+90.)

Durabilité numérique et éco-conception

Architecture d'Entreprise

IV — LE RAYONNEMENT

« Pour qu'un système informatique soit durable, il ne suffit pas qu'il fonctionne longtemps. Il faut aussi qu'il consomme peu, qu'il soit maintenable, et qu'il respecte la vie privée de ses utilisateurs. » — The Shift Project, Lean ICT Report (2019)

« Le numérique n'est pas immatériel. Il a un poids en matériaux, en énergie, et en attention humaine. »

Vidéo d'introduction

 Télécharger cette leçon en PDF

Le serveur qui ne s'éteignait jamais

En 2021, dans une grande banque française, une équipe innovation lance **Green Digital** — un programme d'éco-conception des SI. Au départ, c'est une initiative bottom-up : trois ingénieurs convaincus qu'on peut réduire l'empreinte environnementale du SI sans nuire au métier.

Le premier acte est presque trivial : **mesurer**. L'équipe installe des capteurs sur les 2 800 serveurs du SI, mesure la consommation électrique, la chaleur dissipée, l'utilisation effective. Le résultat est stupéfiant : **40 % des serveurs tournent à moins de 5 % d'utilisation**, 24h/24, depuis des années. Ils consomment de l'énergie pour rien.

Le deuxième acte est plus difficile : **convaincre**. Les directions métiers résistent. « On ne touche pas aux serveurs, disent-elles, on ne sait pas qui les utilise. » L'équipe Green Digital ne baisse pas les bras. Elle propose un protocole : décommissionner un serveur seulement après trois mois d'observation sans impact.

Le troisième acte est le plus utile : **outiller**. L'équipe développe un dashboard d'observabilité environnementale qui montre, application par application, l'empreinte carbone. Les équipes projet commencent à intégrer cette dimension dans leurs décisions : « est-ce que ce nouveau service est vraiment nécessaire, ou est-ce qu'on peut mutualiser avec l'existant ? »

Trois ans plus tard, **Green Digital est devenu un programme groupe**. 600 serveurs décommissionnés, 30 % de réduction de la consommation électrique du SI, et une nouvelle compétence dans l'équipe d'architecture : la capacité à intégrer la soutenabilité dans les décisions.

C'est ce que cette leçon cherche à vous transmettre : **comment la durabilité numérique n'est pas un supplément d'âme, mais une dimension à part entière de l'architecture**.

Le schéma suivant résume les cinq dimensions à mesurer pour évaluer la sobriété d'un SI :

Premier principe — Le numérique n'est pas immatériel

Le numérique consomme des matériaux, de l'énergie, de l'attention humaine. Cette matérialité doit être mesurée, pas ignorée.

Le Shift Project a documenté l'empreinte du numérique :

- **Matériaux** — les serveurs, les data centers, les terminaux nécessitent des métaux rares (lithium, cobalt, terres rares). L'extraction a des conséquences environnementales et sociales.
- **Énergie** — les data centers consomment 1 à 1,5 % de l'électricité mondiale. Cette part est en croissance constante.
- **Attention** — les interfaces numériques captent l'attention humaine, avec des effets documentés sur la santé mentale et la productivité.

L'architecte d'entreprise ne peut pas ignorer ces dimensions. Il a un rôle à jouer dans la réduction de l'empreinte du SI.

La contrefaçon classique consiste à croire que l'éco-conception est un luxe, qu'on s'en occupera « quand on aura le temps ». Cette attitude reporte le problème et le rend plus coûteux. La parade est d'**intégrer la durabilité dès la conception** — comme une exigence non fonctionnelle au même titre que la performance ou la sécurité.

***Et toi ?** Combien de serveurs tournent à moins de 5 % d'utilisation dans ton organisation ? Combien de téraoctets de données sont stockés sans politique de rétention ? Combien de fonctionnalités sont développées puis jamais utilisées ?*

Deuxième principe — L'éco-conception est une exigence non fonctionnelle

L'éco-conception se traite comme une exigence non fonctionnelle : avec des critères mesurables, des seuils à respecter, et des revues à chaque jalon.

Les critères d'éco-conception les plus courants :

- **Efficacité énergétique** — consommation par requête, par utilisateur actif, par téraoctet stocké.
- **Sobriété fonctionnelle** — nombre de fonctionnalités, fréquence d'usage, taux de désuétude.
- **Durabilité technique** — dette technique, dépendance à des bibliothèques non maintenues, complexité cyclomatique.
- **Minimisation des données** — données collectées vs données utilisées, durée de rétention, chiffrement.
- **Accessibilité** — compatibilité avec les terminaux bas de gamme, sobriété des interfaces.

Pour Green Digital, l'équipe a défini cinq indicateurs par application :

- **Score d'efficacité** — consommation / requête
- **Score de sobriété** — fonctionnalités / utilisateurs actifs
- **Score de dette** — warnings SonarQube / KLOC
- **Score de rétention** — téraoctets stockés / téraoctets utilisés
- **Score d'accessibilité** — niveau WCAG atteint

Chaque application a un score agrégé, et un seuil minimal (3/5) en dessous duquel l'application est signalée.

***Et toi ?** Combien de critères d'éco-conception sont définis pour tes applications ? Combien sont mesurés ? Combien ont un seuil à respecter ?*

Troisième principe — La sobriété numérique commence par la minimisation

Le premier principe de l'éco-conception est de ne pas faire ce qui n'est pas nécessaire. La sobriété est plus efficace que l'optimisation.

Trois pratiques de sobriété :

1. **Minimisation fonctionnelle** — ne pas développer ce qui n'est pas nécessaire. Chaque nouvelle fonctionnalité doit avoir un usage documenté et un sponsor identifié.
2. **Minimisation des données** — ne collecter que les données utiles. Le RGPD impose la minimisation comme un principe de licéité (article 5).
3. **Minimisation de la rétention** — supprimer les données obsolètes. Le RGPD impose aussi une durée de conservation proportionnée à la finalité.

La contrefaçon classique consiste à optimiser ce qui est inutile. On optimise un algorithme qui sera de toute façon supprimé. La parade est de **questionner l'existence** de chaque composant, pas son efficacité.

Pour Green Digital, l'équipe a réalisé un **audit de désuétude** sur 87 applications internes. Résultat : 23 applications (26 %) avaient moins de 5 utilisateurs actifs par mois. Ces applications ont été décommissionnées en six mois, libérant 180 té de stockage et 12 serveurs.

***Et toi ?** Combien d'applications dans ton organisation ont moins de 5 utilisateurs actifs par mois ? Combien de téraoctets de données sont collectés mais jamais utilisés ? Combien de fonctionnalités ont été développées sans usage documenté ?*

Quatrième principe — La fin de vie doit être pensée dès la conception

Une application conçue sans stratégie de fin de vie est une dette technique qui se transmet. La conception doit prévoir ce qui se passera quand l'application n'aura plus de valeur.

Trois aspects de la fin de vie :

1. **Décommissionnement** — comment arrêter l'application ? Quelles données migrer ? Quels utilisateurs informer ? Quels contrats arrêter ?
2. **Archivage** — que conserver, sous quelle forme, pour combien de temps ? L'archivage a un coût, et il faut le dimensionner.
3. **Réversibilité** — peut-on revenir en arrière si l'application de remplacement ne fonctionne pas ? La réversibilité a un coût aussi.

La contrefaçon classique consiste à concevoir l'application « pour l'éternité » et à reporter la question de la fin de vie. Quand l'application devient obsolète, personne ne sait comment la décommissionner proprement, et elle reste active pendant des années — consommant des ressources pour rien.

Pour Green Digital, l'équipe a imposé une **clause de fin de vie** dans tous les nouveaux projets : chaque application doit avoir un plan de décommissionnement dès sa conception, avec un budget et un calendrier.

***Et toi ?** Combien d'applications dans ton organisation ont un plan de décommissionnement documenté ? Combien sont en fin de vie sans plan formalisé ? Combien de téraoctets sont archivés « au cas où » depuis plus de 5 ans ?*

Cinquième principe — La sobriété est aussi une question de culture

L'éco-conception ne se décrète pas. Elle se pratique au quotidien, par des milliers de petites décisions. La culture de la sobriété est ce qui rend ces décisions naturelles.

Trois leviers culturels :

1. **La mesure** — ce qui ne se mesure pas ne se gère pas. Mesurer la consommation, la sobriété, la dette technique, et rendre ces mesures visibles.
2. **L'exemplarité** — les architectes et les tech leads doivent montrer l'exemple. Une équipe qui voit son architecte décommissionner un composant inutile sera plus sobre qu'une équipe qui voit son architecte empiler les fonctionnalités.
3. **La récompense** — reconnaître et valoriser les comportements sobres. Pas de prime, mais de la reconnaissance explicite.

Pour Green Digital, l'équipe a tenu un **baromètre mensuel** de la sobriété, affiché dans les locaux de la DSI. Le baromètre montrait les progrès des équipes sur leurs scores d'éco-conception. Les équipes les plus sobres étaient nommées — pas primées, mais reconnues.

Et toi ? Comment l'éco-conception est-elle valorisée dans ton organisation ? Existe-t-il un baromètre visible ? Existe-t-il une reconnaissance explicite des comportements sobres ?

Application — Mon audit de sobriété

À produire séance tenante. Cinq sections précises.

1. **Mesure** — trois indicateurs d'empreinte : consommation électrique, stockage inutilisé, applications sous-utilisées.
2. **Critères** — cinq critères d'éco-conception, notés par application.
3. **Cibles de sobriété** — liste des applications à décommissionner, et des données à supprimer.
4. **Plan de fin de vie** — application pilote, avec un calendrier de décommissionnement.
5. **Engagement culturel** — trois actions pour diffuser la sobriété dans l'équipe.

Cet audit est ton premier livrable de durabilité numérique. Il peut être mené en quelques semaines et produit des gains rapides.

Encadré Au lecteur / À l'animateur

Question d'ouverture : « Combien de serveurs dans ton organisation tournent à moins de 5 % d'utilisation ? »

Profils à observer :

- **L'ignorant**. Ne sait pas combien consomme son SI. *Question* : « Quel est ton dernier chiffre de consommation »

électrique du data center ? »

- **L'optimisateur.** Optimise ce qui est inutile. *Question : « Combien d'applications as-tu décommissionnées cette année ? »*
- **Le collectionneur de données.** Stocke tout « au cas où ». *Question : « Combien de téraoctets n'ont pas été consultés depuis 2 ans ? »*
- **Le fataliste.** Croit que l'éco-conception est un luxe. *Question : « Quel est le coût annuel de ta dette technique ? »*

Pièges d'animation :

- Ne pas laisser la discussion dériver vers la « compensation carbone ». Rappeler que la sobriété est plus efficace que la compensation.
- Ne pas confondre éco-conception et greenwashing. Les actions superficielles ne remplacent pas la transformation profonde.

Ce que la suite vous demande

Maintenant que le cursus est terminé, le chantier continue :

- **Reprendre contact** avec ton sponsor pour présenter les 12 livrables produits pendant le cursus.
- **Prioriser** les 3 chantiers les plus urgents pour ton organisation.
- **Tenir** un comité d'architecture mensuel pour suivre l'avancement.
- **Maintenir** vivante l'architecture : revue annuelle, mises à jour continues.

Le cursus s'arrête ici. La pratique, elle, ne s'arrête jamais.

Sources de cette leçon : The Shift Project, *Lean ICT Report* (2019, 2021) · Règlement Général sur la Protection des Données (RGPD) · ISO 14064 (empreinte carbone des organisations) · Cas Green Digital (anonymisé)

Fin du cursus.

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Quelles sont les trois matérialités du numérique (matériaux, énergie, attention) ? Pourquoi ne doit-on pas les ignorer ?

A2 Pourquoi l'éco-conception doit-elle être traitée comme une exigence non fonctionnelle ? Quels critères peut-on mesurer ?

A3 Qu'est-ce que la sobriété numérique ? Pourquoi la minimisation est-elle plus efficace que l'optimisation ?

A4 Qu'est-ce que la fin de vie d'une application ? Pourquoi doit-elle être pensée dès la conception ?

A5 Quels sont les trois leviers culturels de l'éco-conception (mesure, exemplarité, récompense) ?

A6 Quelle est la part de l'électricité mondiale consommée par les data centers ? Pourquoi cette part est-elle en croissance ?

B — AUTO-EVALUATION

B1 Combien de serveurs tournent à moins de 5% d'utilisation dans ton organisation ? Combien de téraoctets sont stockés sans politique de rétention ?

B2 Combien de critères d'éco-conception sont définis pour tes applications ? Combien sont mesurés ? Combien ont un seuil à respecter ?

B3 Combien d'applications dans ton organisation ont moins de 5 utilisateurs actifs par mois ? Combien de téraoctets collectés mais jamais utilisés ?

B4 Combien d'applications ont un plan de décommissionnement documenté ? Combien sont en fin de vie sans plan formalisé ?

B5 Comment l'éco-conception est-elle valorisée dans ton organisation ? Existe-t-il un baromètre visible ?

B6 Combien de serveurs dans ton organisation tournent à moins de 5% d'utilisation ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Mesurer trois indicateurs : consommation électrique du SI, stockage inutilisé, applications sous-utilisées. (Échéance : J+14.)

C2 Définir cinq critères d'éco-conception et les noter par application. (Échéance : J+30.)

C3 Lister les applications à décommissionner et les données à supprimer, avec un calendrier. (Échéance : J+45.)

C4 Choisir une application pilote et rédiger son plan de décommissionnement. (Échéance : J+60.)

C5 Tenir un baromètre mensuel de la sobriété, visible dans les locaux de la DSI. (Échéance : J+30.)

C6 Reconnaître explicitement les comportements sobres dans l'équipe (sans prime, mais reconnaissance). (Échéance : J+30.)

Le leadership politique de l'architecte

Architecture d'Entreprise

V — LE LEADERSHIP

« Le leadership est l'art de faire faire à quelqu'un ce que vous voulez, parce qu'il a envie de le faire. » — Dwight D. Eisenhower

« L'autorité, c'est quand vous pouvez dire "faites-le parce que je l'ordonne". L'influence, c'est quand vous pouvez dire "faites-le parce que vous me faites confiance, parce que vous voyez que je comprends votre problème, et parce que nous allons tous y gagner". » — Note d'un architecte senior, anonyme, carnet de terrain

Vidéo d'introduction

[Vidéo à produire — Denise Neural, ~90 secondes, manifeste du leadership d'influence]

Le comité de direction qui ne se tient pas

Septembre 2024. Salle de réunion du troisième étage, Mutuelle Plus. Le directeur commercial vient d'éclater : « Vous n'avez pas le droit de bloquer mon projet. Vous êtes là pour dessiner des schémas, pas pour me dire comment vendre. » Face à lui, l'architecte d'entreprise, recruté six mois plus tôt pour « cadrer la transformation du SI ». Il n'a pas de pouvoir hiérarchique sur le commercial. Il n'a pas le budget. Il n'a pas le mandat écrit. Il a, dans sa tablette, la cartographie des processus de vente et les trente-six failles de conformité RGPD que le projet en cours ferait apparaître dans le prochain rapport de l'ACPR. Et il sait que le DAF, à sa droite, a besoin de ce rapport propre pour défendre le ratio de solvabilité.

Que fait-il ?

Il ne sort pas ses schémas. Il ne convoque pas la matrice de Mendelow. Il commence par une phrase, celle qu'il a apprise en observant les négociations commerciales des vingt dernières années : « Pierre, je ne suis pas ici pour bloquer votre projet. Je suis ici pour que vous puissiez le défendre devant l'ACPR dans six mois sans perdre votre bonus. » Pierre se tait. Le DAF hoche la tête. Le tour est joué.

Cette scène est le quotidien réel de l'architecte d'entreprise. Elle n'a rien de technique. Elle est politique, au sens noble : il s'agit de comprendre où sont les intérêts, où sont les leviers, comment amener chacun à faire un pas vers la cible sans avoir l'impression de céder. C'est ce que ce corpus appelle le **leadership politique de l'architecte** — et c'est, selon le retour des experts qui ont pratiqué, **80 % du métier**.

Cette leçon n'enseigne pas une méthode supplémentaire. Elle enseigne ce qui sépare un sachant technique d'un architecte opérationnel. Un sachant connaît TOGAF, ArchiMate, BPMN, les patterns d'EDA, le Data Mesh. Un architecte opérationnel sait **amener une organisation à se transformer en utilisant ces outils sans qu'elle ne se divise**.

1. La posture d'influence : cinq leviers, zéro hiérarchie

L'architecte d'entreprise n'a, par construction, **aucune autorité hiérarchique** sur les directions métiers, les Product Owners, les développeurs, ni même les chefs de projet. Son pouvoir vient exclusivement de sa capacité à **rendre visibles des choses que personne d'autre ne voit** — et à le faire d'une manière qui rend service à chacun.

Cinq leviers composent cette posture. Aucun ne fonctionne seul. C'est leur combinaison qui fait l'architecte influent.

L'expertise comme fondement de légitimité. *On ne peut pas influencer quelqu'un qui vous méprise intellectuellement.* L'AE doit donc d'abord être irréfutable sur le fond : connaître les flux métier critiques (cotisations, prestations, réclamations chez Mutuelle Plus), savoir démontrer l'impact d'un défaut d'architecture sur les délais opérationnels (72 heures pour répondre à un adhérent), connaître les outils périphériques qui, bien que pragmatiques, dégradent la qualité globale. Cette expertise n'est pas une fin en soi ; elle est le **ticket d'entrée** dans la salle où se prennent les décisions.

La coalition stratégique. *On ne convainc pas seul.* Pour briser un silo métier, l'AE cherche des alliés structurels : le contrôleur interne (qui partage la préoccupation de conformité), le RSSI (qui partage la préoccupation de sécurité), le DAF (qui partage la préoccupation de solvabilité). Ensemble, ils transforment des contraintes externes (RGPD, ACPR, ratio de solvabilité) en **leviers internes** pour justifier la transformation. L'AE qui arrive seul à un comité de direction a déjà perdu. L'AE qui arrive avec deux alliés naturels qui reformulent son propos dans le vocabulaire de leur préoccupation a déjà gagné.

La persuasion par la valeur métier. *Le jargon technique est un mur, pas un pont.* L'AE qui parle « Event-Driven » ou « Data Mesh » à un comité de direction parle pour lui-même. L'AE qui traduit ces choix en **TCO**, en **risque de solvabilité**, en **rapidité de mise sur le marché de nouveaux produits** parle à ses interlocuteurs. Le pivot n'est pas de simplifier, c'est de **traduire** — la même idée, dans un vocabulaire qui résonne chez l'autre. Chez Mutuelle Plus, chaque dérogation aux standards rapproche l'entreprise du seuil minimum de solvabilité. L'architecture est le **rempart du capital**, pas un exercice de modélisation.

L'échange et la réciprocité. *La transformation n'est pas un dû.* L'AE qui exige la migration immédiate d'un système qui fait vivre une équipe depuis vingt ans se heurte à une résistance prévisible. L'AE qui propose un **donnant-donnant** (« je numérise vos contrats cette année pour désengorger votre service ; en échange, vous alignez votre prochaine refonte sur le CRM transverse ») obtient l'adhésion. C'est la tactique du *small win* — un gain immédiat, visible, qui prouve la valeur de la coopération.

La contrainte douce, ou les garde-fous. *L'architecte n'ordonne pas, il trace.* Le comité d'architecture n'est pas un tribunal. C'est un lieu où l'on **documente les écarts** par rapport aux standards. Quand un projet dérive, ce n'est pas l'AE qui le sanctionne : c'est le comité qui **constate** l'impact de la dérive sur la solvabilité, la conformité, la dette technique. La traçabilité est une arme douce : elle n'empêche personne de décider, mais elle rend visible le coût de la décision. C'est, à terme, plus persuasif que l'interdiction.

2. La matrice d'influence des parties prenantes : modèle de Mendelow appliqué

Une fois la posture posée, l'AE doit identifier **qui influence quoi**. Tous les acteurs d'une organisation n'ont pas le même poids, ni le même intérêt. Le modèle classique de **Mendelow** (1991) croise pouvoir et intérêt en quatre quadrants. Chaque quadrant appelle une tactique différente.

Quadrant	Acteurs types	Tactique d'influence
Pouvoir élevé / Intérêt élevé (gérer activement)	Directoire, DSI, directeurs de projets structurants	Co-construction : ne jamais présenter une cible finie. Intégrer ces acteurs dans le processus pour qu'ils s'approprient les <i>Building Blocks</i> .
Pouvoir élevé / Intérêt	RSSI, Direction	Rassurance et veto : ils détiennent un pouvoir de blocage. Fournir

faible (satisfaire)	Financière, Risques	des indicateurs de risque clairs, maintenir leur soutien passif.
Pouvoir faible / Intérêt élevé (tenir informé)	Lead Devs, Product Owners, experts métier	Respect de l'expertise : valoriser leur mémoire institutionnelle, vendre la réduction de leur dette technique, éviter le rejet.
Pouvoir faible / Intérêt faible (surveiller)	Support technique, administratifs	Communication minimale : information descendante ciblée pour éviter de nourrir l'inquiétude.

Cette cartographie n'est pas un outil RH. C'est un outil **politique** : elle permet à l'AE de concentrer son énergie sur les **acteurs clés** qui décideront, sans négliger ceux qui bloqueront par inertie ou par hostilité. Elle doit être révisée à chaque phase de l'ADM : un acteur « faible » peut devenir « clé » si la transformation le concerne directement.

Un point souvent oublié : **l'influence informelle est aussi puissante que l'organigramme**. Dans une organisation de 800 personnes comme Mutuelle Plus, où beaucoup d'employés ont grandi avec l'entreprise, un chef d'équipe respecté pèse parfois plus qu'un directeur. Cartographier le pouvoir formel sans cartographier le pouvoir informel, c'est passer à côté de la moitié du jeu.

3. Quatre styles d'animation, à basculer en temps réel

L'atelier d'architecture est l'arène où se jouent les compromis. L'AE n'y dessine pas des schémas — il y **facilite un acte de communication**. Quatre styles s'offrent à lui, et l'art consiste à basculer de l'un à l'autre en quelques secondes selon la dynamique du groupe.

Le style directif (gestion de crise). Quand le débat s'enlise dans les détails, quand le temps presse, quand une décision doit être tranchée pour débloquer un projet. L'AE se lève, occupe l'espace, rappelle fermement les objectifs validés en Phase A. C'est le style du pilote qui reprend les commandes. Il est efficace mais coûteux : il use la confiance s'il est utilisé trop souvent ou trop tôt. À réserver aux moments où l'enlèvement menace l'atelier entier.

Le style persuasif (vente de la cible). Quand il faut embarquer une équipe ou un sponsor. L'AE vend la fin des ressaisies manuelles, la reconquête des jeunes cotisants, la simplicité d'usage des nouveaux outils. Il s'appuie sur des **succès rapides** (*quick wins*) déjà obtenus comme preuves. C'est le style du visionnaire qui sait raconter une histoire.

Le style consultatif (écoute active). Quand une direction métier est en surcharge, quand un conflit latent couve, quand l'AE sent qu'il ne dispose pas de toute l'information. Il laisse les interlocuteurs exprimer leur douleur, leurs contraintes, leurs peurs. Il ne porte pas de jugement. Il absorbe. C'est le style du médecin avant diagnostic — il faut avoir examiné avant de prescrire.

Le style consensuel (validation collective). Quand la décision doit être partagée pour être exécutée. L'AE propose une option qui rallie, identifie les points d'accord, laisse le groupe sceller la décision. C'est le style du consensus nécessaire dans les organisations où les agents seniors n'hésiteront pas à **saboter** un outil qu'ils n'ont pas choisi.

Méthode de bascule en temps réel :

- **Silence dans la salle, attentisme** : basculer du persuasif vers le consultatif. Poser une question ouverte sur le quotidien des participants.
- **Conflit technique bloquant** : sortir du consensuel pour le directif. Trancher en fonction de la trajectoire stratégique globale, documenter la décision.
- **Confusion sur la valeur** : repasser en persuasif en recentrant sur le bénéfice métier immédiat et chiffré.
- **Fatigue collective** : basculer en consensuel pour valider un palier de transition et lever la séance sur une victoire.

L'erreur classique du débutant est de **rester dans son style naturel**. L'architecte efficace est caméléon : il détecte le climat de la salle, il s'adapte. Le style ne trahit pas la personnalité ; il sert l'objectif.

4. Sept objections types et leurs parades

L'architecte d'entreprise qui n'a jamais essayé l'une de ces sept objections n'a probablement pas encore rencontré de directeur métier. Chacune a une **posture d'écoute** sous-jacente (ce que l'interlocuteur dit vs ce qu'il ressent) et une **réponse verbale** adaptée.

« **Vous ne connaissez pas la réalité du terrain.** » (Objection tour d'ivoire) *Posture d'écoute* : besoin de reconnaissance d'une expertise historique. Le directeur a construit son service, il en connaît les failles mieux que personne, et il craint qu'un schéma venu d'en haut ne les ignore. *Réponse* : « Mon architecture ne vaut rien sans votre expertise. Je viens passer deux jours en immersion avec vos gestionnaires pour m'assurer que le système s'adapte à eux, et non l'inverse. »

« **On a des dossiers à traiter, pas le temps pour vos schémas.** » (Manque de temps) *Posture d'écoute* : surcharge opérationnelle réelle, sous-effectif chronique, sentiment que l'AE est un luxe qu'on ne peut pas se permettre. *Réponse* : « Précisément. Si on ne prend pas deux heures pour l'architecture aujourd'hui, vos équipes passeront encore 50 % de leur temps à corriger des erreurs de saisie dans six mois. »

« **L'AE est un luxe qu'on ne peut pas s'offrir.** » (Coût excessif) *Posture d'écoute* : inquiétude sur le budget et le ratio de solvabilité. L'AE est perçu comme une dépense, pas comme un investissement. *Réponse* : « L'architecture n'est pas un coût. C'est l'assurance-vie de notre solvabilité. Elle évite les investissements redondants qui, cumulés sur cinq ans, dépassent de trois fois le coût d'une équipe AE. »

« **Mes informaticiens détestent vos standards.** » (Rejet des développeurs) *Posture d'écoute* : peur de perdre en autonomie technique et en créativité. Les standards sont vécus comme une contrainte bureaucratique. *Réponse* : « Les standards ne sont pas des prisons, ce sont des fondations. On va définir ensemble les briques qui libèrent vos équipes des tâches ingrates pour qu'ils se concentrent sur la valeur métier. »

« **Gest+ tourne depuis vingt ans, ça suffit bien.** » (Absence de besoin) *Posture d'écoute* : nostalgie, déni face à l'obsolescence. Le système est connu, maîtrisé ; tout changement est perçu comme un risque. *Réponse* : « Gest+ est solide pour le monde qu'il servait. Mais il ne parle pas le langage des jeunes de 25 ans. Si on ne change pas, notre base client disparaîtra avec ses retraités. »

« **J'ai déjà acheté mon logiciel, votre avis importe peu.** » (Choix déjà fait) *Posture d'écoute* : tentative de passage en force pour préserver l'autonomie du silo. L'achat est souvent déjà signé. *Réponse* : « Très bien. Comment va-t-il se synchroniser avec le référentiel contrat ? Sans architecture d'intégration, vous créez un nouveau silo qui augmentera vos coûts de support de 20 % par an. »

« **C'est un sujet DSI, ne m'embêtez pas avec ça.** » (Désintérêt business) *Posture d'écoute* : perception de l'IT comme un centre de coût, pas comme un levier stratégique. *Réponse* : « L'architecture est l'outil de votre stratégie. C'est elle qui permettra de sortir vos nouveaux produits en trois mois au lieu de douze. »

Ces réponses ne sont pas des formules magiques. Ce sont des **points de départ** à adapter au contexte. Le débutant les apprendra par cœur ; l'architecte opérationnel les internalise et improvise.

5. Dix phrases pour désamorcer un conflit en atelier

Maintenir la sécurité psychologique d'un atelier est un art. Une seule phrase maladroite peut rompre la confiance patiemment construite. Voici dix phrases éprouvées, à placer quand la tension monte :

1. « *Je perçois une frustration réelle sur [le sujet], et elle est légitime.* »
2. « *Votre connaissance du métier est notre plus grand actif. Ne la gâchons pas en polémiques.* »
3. « *Recentrons-nous : l'objectif est [rappel factuel de l'objectif validé].* »
4. « *Nous divergeons sur l'outil, mais nous sommes d'accord sur le besoin, n'est-ce pas ?* »
5. « *Quelle solution proposez-vous pour respecter [la contrainte externe] sans alourdir votre travail ?* »
6. « *Je note ce point de friction. Nous allons l'isoler pour une étude d'impact dédiée, afin de ne pas bloquer l'atelier.* »

7. « *Mon rôle est de protéger la cohérence globale du SI. Ce qui protège aussi votre futur projet.* »
8. « *Faisons une pause de cinq minutes. La tension actuelle nous empêche d'être constructifs.* »
9. « *Je m'engage à ce que cette contrainte soit arbitrée au plus haut niveau.* »
10. « *Cherchons un palier de transition acceptable, plutôt qu'une cible parfaite impossible à atteindre.* »

Trois principes sous-jacents. **Reconnaître la légitimité** de la frustration (1, 2). **Reformuler le commun** avant de traiter le désaccord (4, 5). **Offrir une sortie honorable** sans céder sur la cible (6, 9, 10). Ces phrases ne sont pas de la manipulation. Elles sont de la **clarification** : elles permettent au groupe de continuer à travailler sans se perdre dans l'émotion.

6. Les trois erreurs fatales

Trois postures tuent la crédibilité d'un architecte plus sûrement que toutes les autres. Elles sont tentantes — surtout pour les profils techniques brillants.

L'usurpation de pouvoir (donner des ordres). L'AE n'est pas chef de projet. Donner des directives aux équipes opérationnelles crée des conflits de territoire immédiats et paralyse la collaboration. « *Votre pouvoir est d'influence, pas de commandement* » — c'est la phrase qui doit rester affichée au mur.

Le dogmatisme technique (se cacher derrière les standards). Utiliser TOGAF, ArchiMate ou BPMN comme une fin en soi est une preuve de faiblesse. Quand un interlocuteur entend « la méthode l'impose », il entend « je ne sais pas justifier la valeur métier du choix ». Le standard doit servir le business, pas l'inverse.

L'arrogance ou l'esquive (feindre de tout savoir). Face à des experts qui ont vingt ans de maison, l'arrogance est suicidaire. Ne masquez pas vos limites sur un processus spécifique. Adoptez la posture de **l'apprenant expert** : vous maîtrisez la méthode et la vision d'ensemble, mais vous avez besoin de leur savoir pour remplir les cases.

Ces trois erreurs ont un point commun : elles confondent **autorité intellectuelle** et **autorité hiérarchique**. La première se gagne, se perd, se regagne. La seconde n'existe pas, en AE, et tenter de l'usurper détruit tout.

7. Quatre principes opérationnels

De cette pratique se dégagent quatre principes. Ils ne sont pas théoriques : ce sont les condensés de centaines d'heures d'atelier, de comités, de conflits désamorçés — ou ratés.

Premier principe — L'influence se gagne, l'autorité s'usurpe. *L'AE n'a pas de mandat coercitif. Il a un mandat d'éclairage.* La différence est fondamentale : on ne peut pas ordonner, on peut seulement convaincre. Conséquence pratique : tout livrable d'architecture doit être **prêt-à-décider** au moment où la décision se prend, pas après. C'est le fameux *decision-ready* cher à TOGAF. Si l'AE arrive après la décision, il ne sert plus à rien.

Deuxième principe — Cartographier le pouvoir avant de cartographier le SI. *Le schéma directeur ne commence pas par les applications, il commence par les acteurs.* Avant de modéliser la cible, l'AE doit avoir cartographié qui décide, qui bloque, qui informe, qui surveille. Sans cette cartographie, la cible technique est un château de cartes : elle s'effondrera au premier conflit politique. Mendelow n'est pas un outil de consultant. C'est un outil de survie.

Troisième principe — Traduire, jamais jargonner. *Le même contenu, deux vocabulaires.* Devant un comité exécutif : TCO, ROI, risque de solvabilité. Devant une équipe technique : patterns, dette, Architecture Runway. Devant un métier : délai, ergonomie, conformité. La forme suit l'audience. L'AE qui ne sait pas traduire est un AE qui ne sera pas compris — et donc pas suivi.

Quatrième principe — Documenter les écarts, jamais les sanctionner. *Le comité d'architecture est un lieu de traçabilité, pas un tribunal.* Quand un projet dérive, l'AE **documente** l'écart, son impact, ses risques. Il ne sanctionne pas. La sanction vient naturellement du fait que, six mois plus tard, quand le projet dérape, le rapport écrit existe, et personne ne peut dire « on ne savait pas ». C'est la **dette politique** : elle se paie, mais à terme, pas à chaud.

8. Application — Votre première cartographie politique

Trois livrables à produire dans les 21 jours suivant la leçon.

Livable 1 — La matrice de votre organisation. Construisez votre propre matrice de Mendelow sur votre périmètre d'AE actuel. Pour chacun des 10-15 acteurs que vous côtoyez réellement (pas les fonctions abstraites), placez-les dans un quadrant. Nommez la tactique que vous avez appliquée — ou que vous auriez dû appliquer. Notez où vous avez surestimé ou sous-estimé le pouvoir informel.

Livable 2 — Traitez trois objections réelles. Identifiez trois objections que vous avez réellement entendues dans les trois derniers mois. Pour chacune, rédigez : - la **posture d'écoute** (ce que l'interlocuteur disait vraiment) ; - la **réponse verbale** que vous avez donnée (notez-la même si elle était médiocre) ; - la **réponse améliorée** que vous donneriez aujourd'hui, en appliquant les principes de cette leçon.

Livable 3 — Un atelier-témoin. Conduisez un atelier d'architecture (même court, même restreint) en pratiquant explicitement les quatre styles. Notez après l'atelier : - quel style vous avez utilisé naturellement ; - quel style vous avez dû forcer ; - quel moment a nécessité un basculement, et pourquoi.

Cette leçon ne transforme pas un sachant en leader. Elle transforme un leader potentiel en leader opérationnel. La différence se voit dans les six mois qui suivent.

Encadré Au lecteur / À l'animateur

Question d'ouverture (à poser à voix haute avant la lecture) : « Racontez-moi une décision importante dans votre organisation récente qui a été prise contre l'avis de la DSI ou de l'équipe d'architecture. Pourquoi a-t-elle été prise ? Qu'est-ce que l'AE aurait pu faire différemment ? »

Profil à observer chez l'apprenant :

- **Le technicien qui refuse la dimension politique.** Considère que « si le schéma est bon, il sera suivi ». *Question : « Quand avez-vous vu un excellent schéma ignoré par un comité ? Pourquoi ? »*
- **Le cynique qui a abandonné l'influence.** A renoncé à convaincre, ne produit plus que des notes techniques pour la postérité. *Question : « Quelle est la dernière fois où vous avez sincèrement essayé de convaincre ? Qu'est-ce qui a échoué ? »*
- **Le dogmatique qui veut imposer TOGAF.** Croit que citer un standard suffit à trancher. *Question : « Traduisez-moi votre dernier principe d'architecture en impact financier pour le DAF. »*
- **Le chef de projet qui se prend pour un AE.** Confond autorité de projet et influence d'architecture. *Question : « Quand vous quittez le projet, que reste-t-il de votre influence ? »*

Pièges d'animation :

- Ne pas laisser la discussion dériver vers « la politique est sale ». Rappeler que l'influence sans autorité est la compétence centrale de l'AE, pas une compromission.
- Ne pas confondre cette leçon avec un cours de management. Elle est strictement outillée pour l'AE, pas pour le directeur général.
- Insister sur le fait que ces techniques sont des **points de départ**, pas des formules magiques. L'architecte opérationnel les adapte, les improvise, les fait siennes.

Si profil d'apprenant ≠ novice : ouvrir sur la tension entre l'AE « pure » (rôle de conseil) et l'AE « politique » (rôle d'influence). Faire expliciter à l'apprenant où il se situe, et quels sont les leviers qu'il n'a pas encore activés.

Ce que le cursus vous demande maintenant

Vous avez maintenant traversé douze leçons techniques — vision, métier, données, applicatif, technique, gouvernance, durabilité. Vous avez acquis les savoirs. Cette treizième leçon vous invite à acquérir le **savoir-faire politique** qui transforme ces savoirs en transformation réelle.

Ouvrez votre dossier d'architecte à une nouvelle page. Préparez, dans les 30 jours :

- La **matrice de Mendelow** de votre propre périmètre, avec une tactique par acteur clé.
- **Trois réponses améliorées** à des objections réelles que vous avez subies.
- Le **récit** d'un atelier où vous avez basculé d'un style d'animation à l'autre en temps réel — et ce que vous en avez appris.

La leçon suivante — *Les Patterns d'Architecture du XXI^e siècle* — traite des **outils** modernes (Event-Driven, Data Mesh, GenAI, FinOps) que vous pourrez désormais choisir, vendre et implémenter en sachant **pourquoi** ils servent votre stratégie — et **comment** les faire accepter par une organisation réelle.

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Pourquoi l'architecture d'entreprise est-elle « 80 % politique » selon le retour des experts ? Que signifie cette affirmation ?

A2 Quels sont les cinq leviers d'influence sans autorité hiérarchique ? En quoi chacun est-il nécessaire ?

A3 Qu'est-ce que la matrice de Mendelow ? Quels sont les quatre quadrants et la tactique associée à chacun ?

A4 Quels sont les quatre styles d'animation d'atelier (directif, persuasif, consultatif, consensuel) ? Quand bascule-t-on de l'un à l'autre ?

A5 Pourquoi la réponse à une objection doit-elle distinguer « posture d'écoute » et « réponse verbale » ? Donnez un exemple concret.

A6 Quelles sont les trois erreurs fatales de posture de l'architecte (usurpation, dogmatisme, arrogance) ? Pourquoi sont-elles fatales ?

B — AUTO-EVALUATION

B1 Sur ton dernier projet d'architecture, as-tu identifié les acteurs clés avant de modéliser la cible ? Combien t'en avaient échappé ?

B2 Combien d'acteurs dans ton périmètre actuel sont dans le quadrant « pouvoir élevé / intérêt faible » ? Leur as-tu fourni les bons indicateurs ?

B3 Sur tes cinq derniers ateliers, combien de fois as-tu dû basculer de style en temps réel ? Quel a été le déclencheur ?

B4 Combien d'objections types (parmi les sept de la leçon) as-tu réellement entendues dans les trois derniers mois ? Lesquelles as-tu mal traitées ?

B5 T'est-il arrivé de confondre autorité intellectuelle et autorité hiérarchique ? Quel a été le coût de cette confusion ?

B6 Combien de fois as-tu accepté un rôle d'exécution alors que tu devais rester en posture d'influence ? Quel est l'effet sur ta crédibilité ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Construire ma matrice de Mendelow sur mon périmètre d'AE actuel (10-15 acteurs réels, pas des fonctions). (Échéance : J+14.)

C2 Identifier trois objections réelles entendues dans les trois derniers mois et rédiger une réponse améliorée pour chacune. (Échéance : J+21.)

C3 Conduire un atelier-témoign en pratiquant explicitement les quatre styles d'animation et noter les basculements. (Échéance : J+30.)

C4 Identifier un acteur clé « pouvoir faible / intérêt élevé » et lui proposer un échange gagnant-gagnant dans les 30 jours. (Échéance : J+30.)

C5 Documenter trois écarts aux standards que j'ai observés sans les sanctionner, avec leur impact chiffré. (Échéance : J+45.)

C6 Préparer une présentation « decision-ready » sur un sujet stratégique où la décision sera prise dans les 60 jours. (Échéance : J+60.)

Les patterns d'architecture du XXIe siècle

Architecture d'Entreprise

V — LE LEADERSHIP

« L'architecture d'entreprise du XXIe siècle ne décrit pas un état stable du système d'information. Elle définit un potentiel de situation — la capacité de l'organisation à pivoter vite, à composer de nouvelles solutions à partir de briques autonomes, à intégrer l'intelligence artificielle sans la subir. » — Note d'un architecte senior, retour d'expérience sur un programme de transformation

« Le coût d'une donnée manquante n'est jamais le coût de la donnée. C'est le coût de la décision que vous n'avez pas pu prendre. » — Loi opérationnelle du data-driven, source anonyme, carnets de terrain

Vidéo d'introduction

[Vidéo à produire — Denise Neural, ~90 secondes, panorama des 4 patterns]

Le ticket de 72 heures

Septembre 2024, plateforme de services de Mutuelle Plus. Un adhérent appelle pour signaler une erreur sur son dernier remboursement. Le conseiller ouvre le dossier. Le dossier s'ouvre sur Gest+, le progiciel métier. Le progiciel cherche dans la base. La base prend huit secondes pour répondre — c'est acceptable. Le conseiller vérifie le remboursement, détecte une anomalie, veut la corriger. La correction nécessite de valider une exception, ce qui exige un workflow dans le module « Workflow » de Gest+. Le module est ralenti par les traitements batch du matin qui n'ont pas encore libéré la base. Le conseiller prend son déjeuner, revient, réessaie, échoue à nouveau. Trois jours plus tard, l'anomalie est corrigée. L'adhérent a déjà résilié.

Ce scénario n'est pas une exception chez Mutuelle Plus. C'est la norme : **72 heures** entre la déclaration d'un problème et sa résolution. Le marché — Alan, Alan, Malakoff Humanis, les nouveaux entrants digitaux — répond en moins de deux heures. Pour une mutuelle de 800 000 personnes protégées, chaque point de pourcentage de résiliation représente huit mille adhérents perdus. À 300 € de cotisation moyenne, c'est 2,4 millions d'euros de chiffre d'affaires volatilisé par an.

L'architecture d'entreprise des années 2010 — un progiciel central, des batchs nocturnes, des interfaces synchrones — n'est plus capable de soutenir la promesse client. Le XXIe siècle impose un changement de paradigme. Ce ne sont pas de nouveaux outils : ce sont de **nouvelles manières de penser le SI**. Quatre patterns redéfinissent ce qu'un architecte doit maîtriser aujourd'hui. Ils ne se substituent pas à TOGAF, BPMN, ArchiMate : ils en sont la **couche moderne**, celle qui permet à la méthode de produire de la valeur métier au rythme du marché.

Cette leçon présente ces quatre patterns — Event-Driven Architecture, Data Mesh, GenAI et FinOps — et montre comment l'architecte les intègre dans un cycle ADM contemporain, sans dogmatisme, avec un objectif clair : transformer la dette en potentiel.

1. Event-Driven Architecture — quand le SI devient un système nerveux

L'architecture événementielle part d'un constat simple : un SI traditionnel traite les demandes **à la demande** (*request-response*). Chaque interaction crée un couplage temporel entre l'émetteur et le récepteur : l'émetteur attend que le récepteur ait fini pour continuer. Quand le récepteur est ralenti, l'émetteur est bloqué. Quand le récepteur tombe, l'émetteur échoue.

L'architecture événementielle inverse la logique. Chaque changement d'état significatif dans le SI devient un **événement**, émis sur un canal, consommé par les acteurs intéressés — sans attendre, sans coupler. C'est la différence entre un appel téléphonique (synchrone, bloquant) et un email (asynchrone, non bloquant) — appliquée à l'échelle du SI.

Cinq composants fondamentaux.

Composant	Rôle	Exemple Mutuelle Plus
Producer	Détecte un changement d'état et émet l'événement	Le module « Sinistre déclaré » de Gest+, le CRM agent, le portail adhérent
Event	Message immuable notifiant un fait accompli	« Sinistre déclaré par l'adhérent X le 12/09 à 14h32, montant estimé 250 € »
Channel	Conduit logique transportant l'événement	Un topic Kafka, une file Azure Service Bus, un flux Kinesis
Processor	Composant consommateur exécutant la logique métier	Module de calcul de remboursement, module de détection de fraude, module CRM
Broker	Pivot gérant la distribution, la persistance, le routage	Kafka, RabbitMQ, AWS EventBridge

Cette décomposition n'est pas qu'organisationnelle : elle est **politique**. Elle permet à l'AE d'attribuer à chaque direction métier la propriété d'un sous-ensemble d'événements, sans exiger qu'elle comprenne le SI global. Le DAF n'a pas besoin de comprendre le calcul de remboursement ; il a besoin que l'événement « Sinistre déclaré » soit émis de manière fiable. C'est cette **asymétrie de connaissance** qui rend l'EDA compatible avec des organisations en silos.

Deux grandes familles se distinguent.

L'**Event Sourcing** persiste chaque changement d'état comme un événement, et reconstruit l'état courant par rejeu des événements. C'est l'approche privilégiée pour les SI où l'**audit** est critique : on peut reconstruire l'état du SI à n'importe quel point du passé. Chez une mutuelle, c'est précieux : en cas de contrôle ACPR, on peut répondre à « quel était l'état du dossier de cet adhérent le 12 mars 2023 à 14h ? » sans dépendre d'une hypothétique sauvegarde.

L'**Event Streaming** traite des flux continus d'événements pour réagir **en temps réel** à des motifs. On ne cherche pas à reconstruire le passé ; on cherche à détecter des patterns dans le présent. Chez Mutuelle Plus, c'est ce qui permet d'identifier une fraude en cours (« trois déclarations de sinistres inhabituelles sur les six dernières heures pour le même numéro de contrat ») ou de signaler un dépassement de plafond de garantie en temps réel.

Ces deux familles ne sont pas mutuellement exclusives. Une architecture EDA mature combine souvent les deux : Event Streaming pour la détection temps réel, Event Sourcing pour l'auditabilité.

L'arbitrage central : remplacer un **ESB traditionnel** (Enterprise Service Bus) par un **Message Broker moderne** (Kafka, Pulsar, NATS). L'ESB traditionnel centralise une logique métier complexe, fragile, propriétaire ; il devient le point de défaillance unique. Le Broker moderne privilégie la scalabilité horizontale : on ajoute des consommateurs sans modifier les producteurs, on gère la backpressure sans bloquer les émetteurs, on garantit la persistance sans coupler les flux. C'est cette dissociation qui permet à Mutuelle Plus d'ajouter une nouvelle campagne de santé en quelques semaines sans toucher au progiciel Gest+.

En termes ArchiMate 4, le flux se modélise ainsi : un **Business Event** (« Customer Order ») déclenche un **Business**

Process (« Contract Creation ») qui réalise un **Business Object** (« Active Policy »), émis via une **Flow Relationship** vers les consommateurs intéressés. Les interfaces entre le legacy (Gest+) et les nouveaux services utilisent des messages standardisés (JSON, Avro) pour assurer l’interopérabilité sans modification intrusive du cœur de système.

2. Data Mesh — quand la donnée devient un produit de domaine

L’architecture d’entreprise des années 2010 concevait la donnée comme un **actif central** à maîtriser par la DSI : un Data Warehouse, un MDM, une équipe de data engineers en position de gatekeeper. Ce modèle a deux limites. Premièrement, il crée un goulot d’étranglement : chaque nouvelle question métier nécessite un projet data, avec une équipe data, dans une file d’attente data. Deuxièmement, il déresponsabilise les métiers : les directions opérationnelles consomment la donnée, mais ne la produisent pas avec la qualité attendue. Le résultat, chez Mutuelle Plus comme ailleurs, est un patrimoine data centralisé mais dégradé : doublons dans Gest+, tableurs en agence, données de santé fragmentées entre quinze systèmes.

Le **Data Mesh**, théorisé par Zhamak Dehghani en 2019, propose un changement de paradigme : la donnée n’est plus un actif centralisé, c’est un **produit distribué** sous la responsabilité des domaines métier.

Quatre principes.

- 1. Domain Ownership.** Chaque domaine métier (Santé Individuelle, Prévoyance Collective, Gestion des Réclamations, etc.) possède ses données. La DSI ne « gère » plus la donnée ; elle fournit l’infrastructure qui permet aux domaines de la gérer eux-mêmes. *Conséquence* : les responsables métier sont responsables de la qualité à la source, pas les data engineers en bout de chaîne.
- 2. Data as a Product.** Chaque ensemble de données exposé par un domaine est un « produit » avec un propriétaire nommé, un contrat d’interface (schema, SLA, latence), une documentation utilisateur. *Conséquence* : un produit data est aussi adressable qu’une API ; il peut être consommé par d’autres domaines sans négociation bilatérale.
- 3. Self-Serve Platform.** L’infrastructure data (pipelines, catalogue, gouvernance, monitoring) est fournie comme une plateforme interne que les domaines utilisent sans expertise technique spécialisée. *Conséquence* : les domaines publient et consomment de la donnée sans dépendre d’une équipe data centrale.
- 4. Federated Governance.** Les standards d’interopérabilité (formats, identifiants, règles de sécurité, conformité RGPD) sont définis globalement et appliqués automatiquement. *Conséquence* : la cohérence globale est garantie par la plateforme, pas par un comité manuel.

Cette architecture produit un résultat stratégique majeur : **résolution de la dette d’intégration par les Data Contracts**. Quand un domaine modifie la structure de ses données, le contrat de données expose le changement, et les consommateurs peuvent s’adapter sans casser. C’est la résolution directe du problème que Gest+ connaît aujourd’hui : modifier un champ métier dans Gest+ nécessite de modifier quinze rapports en aval.

Matrice de décision : quand adopter le Data Mesh, quand conserver un MDM centralisé ?

Critère	MDM centralisé	Data Mesh
Maturité data de l’organisation	Moyenne, IT-centric	Haute, culture de domaine établie
Nombre de domaines métier	Faible (< 5)	Élevé (> 10)
Besoin d’agilité	Standard	Critique (transformation continue)
Compétences data disponibles	Centralisées	Distribuées dans les domaines
Culture d’ownership	Faible	Forte

Mutuelle Plus, avec ses quinze domaines opérationnels et son projet de refonte du SI Décisionnel, est aujourd'hui à la **frontière** : un MDM centralisé pourrait encore tenir trois à cinq ans, mais un Data Mesh serait préférable pour soutenir la transformation à cinq ans. L'AE recommande une trajectoire **hybride** : MDM centralisé sur le domaine « contrats » (haute criticité, faible maturité), Data Mesh sur les domaines périphériques (marketing, satisfaction, observatoire).

3. GenAI dans la cartographie applicative — l'architecte augmenté

L'intelligence artificielle générative — Large Language Models, Retrieval-Augmented Generation, agents autonomes — n'est pas un gadget pour l'architecte d'entreprise. C'est un **accélérateur de productivité documentaire** et un **levier de souveraineté sur le patrimoine d'architecture**.

Concrètement, un LLM couplé à un RAG sur le référentiel d'architecture permet :

- **L'analyse d'impact en langage naturel.** Poser une question du type « quelles sont les applications impactées par la migration du module Auth de Gest+ ? » et obtenir une réponse structurée avec les dépendances identifiées. Là où l'AE passait deux jours à feuilleter le référentiel, il obtient une réponse en deux minutes — qu'il valide ensuite par revue humaine.
- **La détection de redondances fonctionnelles.** Soumettre au LLM la liste des applications métiers pour identifier les recouvrements : « Ces deux applications font probablement la même chose, à vérifier avec les métiers. » Mutuelle Plus a ainsi identifié trois doublons fonctionnels entre Gest+ et des outils locaux développés par les services opérationnels.
- **La génération de fiches applicatives.** À partir de la documentation existante (souvent parcellaire, désordonnée), le LLM produit une fiche normalisée : description, fonctionnalités, dépendances, propriétaire, contraintes. L'AE économise huit à quinze jours-homme par cycle d'enrichissement du référentiel.
- **L'assistance à la modélisation ArchiMate.** Demander au LLM de produire un draft de diagramme ArchiMate à partir d'un brief métier en langage naturel. Le résultat est à 70 % exploitable ; les 30 % restants sont à reprendre pour la cohérence stylistique.

Ces gains sont réels. Mais ils viennent avec **trois risques structurels** que l'AE doit encadrer.

Risque 1 — Confidentialité. Un LLM mal configuré peut exposer des données de santé (catégorie spéciale RGPD) dans ses réponses, ou les transmettre à des serveurs tiers. *Atténuation* : modèles privés isolés (Azure OpenAI en mode privé, Mistral self-hosted), hébergement des données du référentiel dans un périmètre cloisonné, audit des flux par le RSSI.

Risque 2 — Hallucinations. Un LLM peut produire des schémas techniques erronés, inventer des dépendances qui n'existent pas, ou proposer des solutions non conformes aux standards internes. *Atténuation* : validation humaine obligatoire par un architecte senior, journalisation des suggestions, mesure de la précision sur un échantillon de validation avant déploiement.

Risque 3 — Dette technique IA. Le code généré, les configurations générées, les schémas générés peuvent être non maintenables, inintelligibles, ou non conformes aux standards de développement. *Atténuation* : standards de codage stricts appliqués au code généré, revue systématique par un Lead Dev, gouvernance par l'Architecture Review Board.

Cadres de gouvernance à appliquer. Le **NIST AI Risk Management Framework** propose quatre fonctions (Govern, Map, Measure, Manage) pour évaluer et mitiger les risques. L'**EU AI Act** (entrée en application progressive depuis 2024) classe les usages par niveau de risque et impose des obligations proportionnelles : un usage en « risque limité » (assistant documentaire) requiert transparence ; un usage en « risque élevé » (décision automatique sur dossier adhérent) requiert certification, auditabilité, supervision humaine.

Pour Mutuelle Plus, l'usage de GenAI en documentation d'architecture relève du **risque limité** sous réserve d'encadrement : données hébergées en privé, validation humaine systématique, audit annuel. Un usage futur pour la détection de fraude ou l'aide à la décision sur les dossiers rentrerait dans le **risque élevé**, avec des obligations sensiblement plus lourdes.

4. FinOps — quand l’efficacité financière devient une exigence d’architecture

Le FinOps — contraction de « Finance » et « DevOps » — est l’ensemble des pratiques qui permettent d’**optimiser en continu les coûts Cloud** en impliquant équipes techniques, financières et métiers dans une gouvernance partagée. Ce n’est pas de la réduction budgétaire : c’est de l’**allocation intelligente** des ressources pour maximiser la valeur par euro investi.

Trois phases structurent la maturité FinOps d’une organisation.

Phase 1 — Inform. Rendre visibles les coûts Cloud par domaine métier, par application, par équipe. Sans cette visibilité, aucune optimisation rationnelle n’est possible. Concrètement : dashboards de cost allocation, tags par domaine, intégration des données de facturation dans le référentiel d’architecture.

Phase 2 — Optimize. Identifier les ressources sous-utilisées, négocier les engagements (Reserved Instances, Savings Plans), ajuster les architectures pour réduire le gaspillage. L’AE intervient ici en challengeant les choix de sizing dans les revues d’architecture : « *Cette instance est dimensionnée pour 10 000 utilisateurs ; tu en as 800. Pourquoi ?* »

Phase 3 — Operate. Intégrer la dimension financière dans la gouvernance continue : budgets par équipe, alertes de dépassement, arbitrages réguliers entre valeur métier et coût d’infrastructure. C’est l’état mature où l’efficacité financière est un réflexe d’équipe.

Trois indicateurs clés pour l’architecte.

KPI	Description	Impact stratégique
Coût par transaction	Coût Cloud par opération métier (remboursement, adhésion, réclamation)	Si le coût IT dépasse la marge des contrats, le modèle économique s’effondre
Idle ratio	Part des ressources provisionnées mais non consommées	Réduction directe du gaspillage des fonds propres
Commitment discount	Remises obtenues par engagement de durée	Optimisation de la capacité d’investissement

Intégration dans l’Architecture Review Board. Chaque design cible présenté à l’ARB doit désormais inclure une checklist financière :

- **Unit Economics** : le coût marginal IT par nouvel adhérent est-il inférieur au coût d’acquisition client ?
- **Cloud Exit Strategy** : existe-t-il un plan de réversibilité documenté pour éviter le verrouillage tarifaire (vendor lock-in) ?
- **Élasticité** : le design permet-il une extinction automatique des ressources hors heures ouvrées ?
- **Coût de la dette** : quelle est la part du coût courant imputable à la dette technique ?

Ces questions ne sont pas optionnelles. Elles transforment l’ARB d’un comité technique en un **comité d’arbitrage stratégique**, où les décisions intègrent simultanément la cohérence architecturale, la conformité, et l’efficacité économique.

5. L’entreprise fractale — la vision cible à cinq ans

Les quatre patterns précédents ne sont pas indépendants. Ils se renforcent mutuellement dans une vision cible que l’on peut qualifier d’**entreprise fractale** : une organisation où l’agilité globale se reproduit à toutes les échelles — du

portefeuille d'affaires jusqu'au composant technique.

Dans cette vision à cinq ans, Mutuelle Plus n'est plus un progiciel central entouré de satellites. C'est un **écosystème de produits** :

- **Couche réactive (EDA)** : tous les événements métier significatifs sont émis en temps réel. Un sinistre déclaré déclenche en parallèle le calcul de remboursement, la détection de fraude, la mise à jour du CRM, la notification à l'adhérent, l'enregistrement comptable. Les traitements batch disparaissent.
- **Couche data distribuée (Data Mesh)** : chaque domaine (Santé Individuelle, Prévoyance, Réclamations, Marketing) expose ses données comme un produit, avec un contrat d'interface et une SLA. Les doublons disparaissent. Les délais d'accès à la donnée chutent de six mois à trois semaines.
- **Couche cognitive (GenAI)** : l'architecte dialogue avec le référentiel en langage naturel. Les nouveaux produits sont qualifiés en deux semaines au lieu de six mois. La documentation est vivante.
- **Couche efficiente (FinOps)** : chaque euro Cloud est tracé, alloué, optimisé. La marge de solvabilité est restaurée. Les réinvestissements dans la transformation sont autofinancés par les gains de productivité.

Cette vision n'est pas un slogan. C'est ce qui permet à une mutuelle de 800 000 adhérents de **lancer une nouvelle garantie santé en six semaines** au lieu de douze mois, avec une marge de solvabilité sécurisée, une conformité RGPD garantie, et une expérience adhérent en temps réel.

L'architecte d'entreprise est le pilote de cette transformation. Il ne dessine pas le SI cible ; il **établit les conditions** pour que l'organisation se transforme elle-même, à son rythme, avec ses contraintes, en s'appuyant sur les quatre patterns comme briques modulaires.

6. Cinq principes opérationnels pour l'architecte du XXI^e siècle

De ces quatre patterns se dégagent cinq principes. Ils ne remplacent pas ceux des leçons précédentes ; ils les complètent pour le contexte contemporain.

Premier principe — Le potentiel de situation prime sur l'état stable. *Ne pas viser une cible figée, viser une capacité d'évolution.* L'AE ne cherche plus à figer l'architecture cible ; il cherche à maximiser la **capacité de l'organisation à pivoter**. L'EDA et le Data Mesh sont des patterns qui produisent du potentiel. Le MDM centralisé et l'ESB synchrones produisent de la rigidité. La question n'est plus « quelle est la cible ? » mais « quelle est la vitesse de réorientation ? »

Deuxième principe — Découpler temporellement, coupler sémantiquement. *Le bon couplage n'est pas l'absence de couplage, c'est le couplage intelligent.* L'EDA découple temporellement (Producteur et Consommateur ne s'attendent pas) mais couple sémantiquement (ils partagent le format de l'événement, le contrat d'interface). Le Data Mesh découple organisationnellement (chaque domaine est autonome) mais couple contractuellement (Data Contracts standardisés). L'AE conçoit les contrats, pas les composants.

Troisième principe — Distribuer la responsabilité, centraliser la gouvernance. *Le Data Mesh n'est pas l'anarchie data.* Chaque domaine est propriétaire de ses produits data, mais la plateforme fédérale garantit les standards d'interopérabilité, de sécurité, de conformité. La gouvernance est globale ; l'exécution est locale. C'est l'application du principe de subsidiarité à l'architecture.

Quatrième principe — Instrumenter avant d'optimiser. *On ne optimise pas ce qu'on ne mesure pas.* Avant d'optimiser les coûts Cloud (FinOps), il faut les mesurer (Phase 1 Inform). Avant d'optimiser les workflows EDA, il faut observer les patterns d'émission et de consommation. Avant de détecter les redondances par GenAI, il faut disposer d'un référentiel d'architecture propre. L'AE qui saute à l'optimisation sans instrumentation crée de la complexité sans valeur.

Cinquième principe — L'IA augmente l'architecte, ne le remplace pas. *Le LLM est un copilote, pas un pilote.* L'AE qui délègue ses jugements à un modèle prend des décisions qu'il ne peut plus justifier. L'AE qui utilise le modèle comme assistant de production — pour générer des drafts, détecter des patterns, suggérer des options — garde la maîtrise de la décision finale. La validation humaine n'est pas une perte de temps ; c'est le **garde-fou de la responsabilité**.

7. Application — Quel pattern pour quel problème ?

Trois exercices à conduire dans les 30 jours.

Exercice 1 — Diagnostic de votre dette d'architecture contemporaine. Pour chacun des quatre patterns, notez votre niveau de maturité actuel sur une échelle de 0 à 3 (0 = absent, 1 = embryonnaire, 2 = en cours, 3 = industrialisé).

- EDA : votre organisation émet-elle des événements métier en temps réel ? Lesquels ? Quel broker ?
- Data Mesh : vos domaines métier possèdent-ils leurs données ? Existe-t-il des Data Contracts formalisés ?
- GenAI : utilisez-vous un LLM pour augmenter votre productivité d'architecte ? Avec quelles garde-fous ?
- FinOps : connaissez-vous le coût par transaction métier de votre SI ? Savez-vous qui consomme quoi ?

Exercice 2 — Choisir un pattern prioritaire. Sélectionnez le pattern où votre dette est la plus critique. Décrivez en une page : le contexte (quel problème métier ?), la cible à 12 mois (qu'est-ce qui aurait changé ?), les trois premières étapes (par quoi commencer ?), le critère de succès mesurable.

Exercice 3 — Cartographier un flux EDA. Choisissez un processus métier réel de votre organisation. Modélisez-le sous forme événementielle : quels sont les producteurs, les événements significatifs, les canaux, les consommateurs, le broker. Comparez avec l'architecture actuelle (synchrone, batch, ESB). Chiffrez le gain de latence attendu.

Cette leçon n'enseigne pas à devenir ingénieur Kafka ou data scientist. Elle enseigne à l'architecte à dialoguer avec ces expertises, à en comprendre les apports et les limites, à arbitrer entre les patterns en fonction du contexte.

Encadré Au lecteur / À l'animateur

Question d'ouverture : « Dans votre organisation, quel est le délai entre un événement métier (commande, sinistre, adhésion) et sa prise en compte par tous les systèmes qui devraient le connaître ? Pourquoi est-il si long ? »

Profils à observer chez l'apprenant :

- **Le technophile qui veut tout réécrire en Kafka.** Confond modernisation et disruption. *Question :* « Quel est le coût de migration de votre ESB vers un broker moderne, et combien de projets métier cela retarderait-il ? »
- **Le conservateur qui refuse les patterns modernes.** Considère que TOGAF et BPMN suffisent. *Question :* « Comment votre organisation lance-t-elle un nouveau produit aujourd'hui ? Combien de temps ? Pourquoi ? »
- **Le data-centrique qui croit que la donnée résout tout.** Veut centraliser toute la donnée avant de la distribuer. *Question :* « Combien de temps avez-vous passé sur votre dernier projet data centralisé ? Quel est le ROI ? »
- **L'IA-enthousiaste qui délègue tout au LLM.** Considère que l'IA va remplacer l'architecte. *Question :* « Quel jugement architectural pouvez-vous déléguer sans perdre votre responsabilité ? »

Pièges d'animation :

- Ne pas laisser la discussion dériver vers les outils (Kafka vs RabbitMQ, OpenAI vs Mistral). Les patterns sont neutres vis-à-vis de l'implémentation.
- Ne pas idéaliser les patterns. Tous ont des coûts, des limites, des prérequis organisationnels. Un EDA mal gouvernée produit plus de chaos qu'une SOA bien gérée.
- Rappeler que ces patterns sont complémentaires, pas substitutifs. Le Data Mesh ne remplace pas le MDM dans tous les cas ; le FinOps ne remplace pas la discipline budgétaire.

Si profil d'apprenant ≠ novice : ouvrir la discussion sur les patterns non couverts ici (architectures cellulaires,

plateformes de données unifiées, edge computing, IA agentique). Identifier le pattern émergent que l'apprenant considère comme stratégique pour son contexte, et l'aider à l'analyser avec la grille de cette leçon.

Ce que le cursus vous demande maintenant

Vous avez maintenant traversé quatorze leçons — douze leçons techniques plus deux leçons sur le leadership et la modernité. Vous avez les savoirs (les frameworks), les savoir-faire (la méthode, les patterns), et les savoir-être (la posture politique). Le cursus s'arrête ici sur les fondamentaux. La suite est votre propre trajectoire.

Trois engagements pour les 60 prochains jours :

- **Cartographier** la maturité de votre organisation sur les quatre patterns de cette leçon.
- **Choisir** un pattern prioritaire et rédiger la première page d'un dossier de modernisation.
- **Initier** une conversation GenAI structurée avec votre DSI ou votre RSSI, sur un cas d'usage limité et encadré.

L'architecture d'entreprise du XXIe siècle n'est pas une discipline figée. C'est une **conversation continue** entre l'organisation et son écosystème technologique. L'architecte est celui qui tient la conversation, qui la structure, qui la fait avancer — sans jamais prétendre en connaître le dernier mot.

Quiz — 18 questions pour valider votre compréhension

Trois séries de six questions. Prenez un crayon, écrivez vos réponses dans l'espace prévu. Score indicatif : 14/18 minimum pour valider la leçon.

A — COMPREHENSION

A1 Quels sont les cinq composants fondamentaux d'une architecture événementielle (EDA) ? Quel rôle joue le broker ?

A2 Quelle est la différence entre Event Sourcing et Event Streaming ? Dans quels cas d'usage privilégier chacun ?

A3 Quels sont les quatre principes du Data Mesh ? Pourquoi distinguent-ils ownership et gouvernance ?

A4 Quels sont les trois risques majeurs de l'usage de GenAI dans la cartographie applicative ? Quelles atténuations proposer ?

A5 Quelles sont les trois phases de maturité FinOps ? Quel est le rôle de l'architecte dans chacune ?

A6 Qu'est-ce qu'une « entreprise fractale » ? En quoi cette vision cible diffère-t-elle d'une architecture cible traditionnelle ?

B — AUTO-EVALUATION

B1 Dans ton organisation, quels sont les événements métier émis en temps réel aujourd'hui ? Lesquels sont encore en batch nocturne ?

B2 Combien de domaines métier dans ton organisation possèdent réellement leurs données ? Combien sont en accès libre via un catalogue ?

B3 As-tu déjà testé un LLM sur ton référentiel d'architecture ? Quel gain de temps as-tu observé ? Quelles erreurs as-tu détectées ?

B4 Connais-tu le coût par transaction métier de ton SI ? Le coût par adhérent / client / usager ? Comment est-il mesuré ?

B5 Combien de fois as-tu été bloqué par un couplage temporel fort (Producteur qui attend Consommateur) dans les six derniers mois ?

B6 Combien de jours sépare, dans ton organisation, un événement métier (commande, sinistre, adhésion) de sa prise en compte par tous les systèmes ?

C — ENGAGEMENTS A 30/45/60 JOURS

C1 Cartographier ma maturité sur les quatre patterns (EDA, Data Mesh, GenAI, FinOps) sur une échelle de 0 à 3. (Échéance : J+14.)

C2 Choisir le pattern où ma dette est la plus critique et rédiger une première page de dossier de modernisation (contexte, cible 12 mois, 3 premières étapes, KPI). (Échéance : J+30.)

C3 Modéliser un processus métier réel sous forme événementielle (5 composants EDA) et chiffrer le gain de latence attendu. (Échéance : J+30.)

C4 Identifier trois domaines métier candidats au Data Mesh et trois domaines où un MDM centralisé reste justifié. (Échéance : J+30.)

C5 Initier une conversation structurée avec mon DSI / RSSI sur un cas d'usage GenAI limité et encadré (données, validation, audit). (Échéance : J+45.)

C6 Mettre en place un dashboard de cost allocation Cloud par domaine métier dans les 60 jours (FinOps Phase 1). (Échéance : J+60.)
