ArchiMate : mes notes
Guide pratique pour débuter en modélisation d'architecture d'entreprise

Stéphane FOSSE
fosse.fr

11 février 2026

Copyleft : cette œuvre est libre, vous pouvez la copier, la diffuser et la modifier
selon les termes de la Licence Art Libre

ArchiMate est un langage de modélisation d'architecture d'entreprise, ouvert et indépendant, maintenu par The Open Group. Il permet de décrire et de relier les couches métier, applicative et technologique d'une organisation dans un formalisme commun. Cette page rassemble mes notes d'utilisation au fil du temps : composants, relations, propriétés, et surtout retours de terrain avec le logiciel libre Archi. Elle propose une ressource en français, pragmatique, pour un architecte qui veut démarrer.

Qu'est-ce qu'ArchiMate et pourquoi l'utiliser ?

ArchiMate se distingue d'UML et de BPMN par son périmètre. Là où UML cible la conception logicielle et BPMN la modélisation de processus métier, ArchiMate embrasse l'ensemble de l'architecture d'entreprise. Le langage a été conçu volontairement restreint : environ cinquante concepts dans sa version 3.2, contre cent cinquante pour UML et deux cent cinquante pour BPMN. Ce minimalisme est un choix de conception. ArchiMate couvre les 80% de cas pratiques sans noyer l'architecte dans un catalogue d'éléments rarement utilisés.

Le langage s'organise en couches : Strategy, Business, Application et Technology. Chaque couche contient des éléments actifs (les acteurs, les composants), des comportements (les processus, les fonctions, les services) et des éléments passifs (les objets métier, les données). Cette grille simple structure la pensée. On sait immédiatement à quel niveau d'abstraction on se situe.

ArchiMate est aligné avec TOGAF, le framework d'architecture d'entreprise de The Open Group, mais il reste utilisable indépendamment. Je l'utilise avec Archi, un outil open source développé depuis 2010 par Phil Beauvoir, disponible sur Windows, macOS et Linux, et téléchargé environ 6 000 fois par mois. Pour quelqu'un qui vient de la culture du logiciel libre, c'est un point d'entrée naturel.

Modèle versus diagramme : une distinction essentielle

C'est probablement la première chose à comprendre, et celle qui change la manière de travailler. Un diagramme Visio ou Draw.io est un dessin. On place des boîtes, on tire des flèches, c'est visuel et c'est tout. Si on renomme une boîte dans un diagramme, l'autre diagramme où elle apparaît ne change pas. On maintient des copies, et ces copies divergent.

Un modèle ArchiMate fonctionne différemment. Les composants existent dans un référentiel central. Les diagrammes – qu'ArchiMate appelle des *vues* – ne sont que des fenêtres sur ce référentiel. Un même Application Component peut apparaître dans dix vues différentes sans jamais être dupliqué. Si on modifie son nom ou ses propriétés, le changement se propage partout. C'est la différence entre dessiner une architecture et la modéliser.

Cette approche rend possible l'analyse d'impact. On peut répondre à des questions comme : « quels processus métier sont affectés si ce composant applicatif est retiré ? » ou « quelles technologies supportent tel service métier ? ». Un diagramme ne répondra jamais à ces questions. Un modèle, si.

Par quoi commencer : l'approche itérative

La tentation est grande de vouloir tout modéliser d'un coup, de la stratégie aux serveurs. C'est le meilleur moyen de ne jamais finir et de décourager les parties prenantes. Mon conseil : commencer petit, commencer concret.

La couche Application est souvent le point de départ le plus naturel pour un architecte SI. On connaît les systèmes, on sait ce qu'ils font, on peut les nommer. Trois types de composants suffisent pour une première vue : **Application Component** (le système), **Application Service** (ce qu'il expose) et **Application Interface** (le point d'accès, une API REST par exemple). Avec ces trois éléments et les relations Assigned To, Realizes et Serves, on peut déjà produire une cartographie applicative utile.

Ensuite, on enrichit. On relie les composants applicatifs aux processus métier de la couche Business. On descend vers la couche Technology pour documenter l'infrastructure. On remonte vers la couche Strategy si on a besoin de relier l'architecture aux capacités et objectifs de l'entreprise. Chaque itération ajoute de la valeur au modèle sans remettre en cause ce qui existe déjà.

Les couches et leurs composants

Couche Strategy

La couche Strategy modélise ce que l'entreprise veut être et comment elle compte y arriver. Une **Resource** représente un actif détenu par l'organisation : ressources humaines, capital financier, propriété intellectuelle. Une **Capability** décrit une capacité que possède l'organisation, par exemple la capacité de gestion de la relation client ou la capacité d'analyse de données. Le **Course of Action** est un plan qui mobilise des ressources et des capacités pour atteindre un objectif, comme une initiative de transformation digitale ou un programme de réduction des coûts.

L'**Outcome** représente un résultat atteint : une augmentation de 20% de la satisfaction client, une position de leader sur un marché, la conformité réglementaire obtenue. Le **Value Stream** est une séquence d'activités qui crée un résultat global pour un client ou une partie prenante, comme le flux *order-to-cash* ou *hire-to-retire*.

Les relations typiques de cette couche suivent une logique de réalisation : Resource réalise Capability, Capability réalise Course of Action, Course of Action réalise Outcome. La Capability sert le Value Stream, et le Value Stream réalise l'Outcome.

Couche Business

C'est la couche la plus riche en nombre de composants, et souvent la plus délicate à modéliser parce que les concepts métier sont moins tangibles qu'un serveur ou une API.

Le **Business Actor** est une entité capable d'agir : un client, un fournisseur, un département marketing. Le **Business Role** représente la responsabilité d'un acteur dans un contexte donné : gestionnaire de comptes, approbateur d'achats, représentant du service client. Un Business Actor est assigné à un Business Role.

Le **Business Process** est une séquence de comportements métier qui produit un résultat spécifique, comme le processus de traitement des commandes ou le processus de recrutement. La **Business Function** regroupe des comportements métier par domaine de compétence : fonction ventes, fonction ressources humaines, fonction contrôle de gestion. La distinction entre les deux est parfois floue ; en pratique, le processus est transversal (il traverse des fonctions) tandis que la fonction est un regroupement organisationnel.

Le **Business Service** est un comportement métier explicitement exposé : service d'intégration client, service de paiement de factures, service de consultation du catalogue produit. Le Business Process réalise le Business Service. Le **Business Object** est un concept manipulé dans le domaine métier : bon de commande, profil client, police d'assurance.

Viennent ensuite le **Business Event** (un changement d'état qui déclenche un processus, comme la réception d'une réclamation client), la **Business Interaction** (un comportement collectif entre plusieurs rôles, comme une négociation de contrat), la **Business Interface** (un point d'accès au service métier : portail client, centre d'appels, magasin physique), le **Product** (une collection cohérente de services et d'objets proposée aux clients : un compte épargne, un package d'assurance, une offre SaaS) et le **Contract** (un accord entre fournisseur et consommateur : SLA, contrat de travail, conditions d'abonnement).

Couche Application

L'**Application Component** est l'élément central : une encapsulation de fonctionnalité applicative, modulaire et remplaçable. Un CRM, une passerelle de paiement, un système de gestion documentaire. L'**Application Service** est le comportement exposé par ce composant : service de traitement des paiements, service de génération de documents, service d'authentification. L'**Application Function** est le comportement interne automatisé : validation de données, génération de rapports, calcul de stocks.

L'**Application Interface** est le point d'accès technique : API REST, interface de service web, interface de connexion à la base de données. Le **Data Object** est la donnée structurée pour le traitement automatisé : enregistrement client, journal de transactions, entrée de catalogue produit. L'**Application Process** est une

séquence de comportements applicatifs : processus automatisé de validation de commande, processus batch de génération de factures.

L'**Application Event** signale un changement d'état applicatif : mise à jour de base de données terminée, upload de fichier achevé, session expirée. L'**Application Collaboration** agrège des composants qui travaillent ensemble : une plateforme e-commerce composée d'un serveur web, d'une base de données et d'un système de paiement. L'**Application Interaction** est le comportement collectif entre composants : échange de données entre CRM et ERP, flux d'authentification SSO.

Couche Technology

Le **Node** est une ressource de calcul ou physique qui héberge d'autres ressources : serveur d'application, serveur de base de données, poste de travail. Le **Device** est la ressource physique concrète : serveur physique, baie de stockage, routeur réseau. Le **System Software** est le logiciel d'infrastructure : système d'exploitation Linux ou Windows Server, SGBD PostgreSQL ou Oracle, serveur d'applications Tomcat ou WebSphere.

La **Technology Interface** est le point d'accès technique : port réseau, interface USB, endpoint API. Le **Path** est un lien entre nœuds : fibre optique, connexion sans fil, tunnel VPN. Le **Communication Network** est un ensemble de structures qui connectent les nœuds : réseau local LAN, réseau étendu WAN, connexion Internet.

Le **Technology Service** est le comportement technologique exposé : service de stockage de données, service de connectivité réseau, service de sauvegarde et restauration. La **Technology Function** regroupe des comportements qu'un nœud peut effectuer : chiffrement de données, répartition de charge, routage réseau. Le **Technology Process** est une séquence technique : processus automatisé de sauvegarde, processus de provisionnement système, processus de déploiement de correctifs de sécurité.

L'**Artifact** est un élément produit ou consommé par un système : fichier de base de données, fichier de configuration, image Docker. La **Technology Collaboration** agrège des nœuds qui coopèrent : cluster haute disponibilité, ferme de serveurs avec répartition de charge, système de stockage distribué. La **Technology Interaction** est le comportement collectif entre nœuds : réplication de base de données, synchronisation de cluster, mécanisme de basculement.

Les relations entre composants

ArchiMate définit un jeu limité de relations qui connectent les composants entre eux et entre les couches. C'est l'un des points forts du langage : les relations sont typées et contrôlées. On ne peut pas connecter n'importe quoi à n'importe quoi. Archi empêche d'ailleurs la création de relations invalides, ce qui guide l'apprentissage.

Relations structurelles

La relation **Assigned To** relie un élément actif à un comportement : un Business Actor est assigné à un Business Role, un Application Component est assigné à une Application Function, un Node est assigné à une Technology Function. C'est la relation « qui fait quoi ».

La relation **Aggregation** indique qu'un élément est composé d'autres éléments : un Product agrège des Business Services, une Application Collaboration agrège des Application Components, une Technology Collaboration agrège des Nodes.

La relation **Composition** est similaire à l'agrégation mais plus forte : l'élément composant ne peut pas exister indépendamment de son conteneur.

Relations de dépendance

La relation **Serves** indique qu'un élément fournit sa fonctionnalité à un autre : un Business Service sert un Business Role, un Technology Service sert un Application Component. C'est l'orientation service du langage qui transparaît ici.

La relation **Realizes** connecte un élément concret à un élément plus abstrait : un Business Process réalise un Business Service, un Application Function réalise un Application Service, un Technology Function réalise un Technology Service. C'est aussi cette relation qui traverse les couches : un Application Service réalise un Business Service, un Technology Service réalise un Application Service.

La relation **Accesses** indique qu'un comportement lit, écrit ou modifie un élément passif : un Business Process accède à un Business Object, une Application Function accède à un Data Object.

Relations dynamiques

La relation **Triggers** modélise une dépendance causale ou temporelle : un Business Event déclenche un Business Process, un Application Event déclenche un Application Process.

La relation **Flow** représente un transfert de quelque chose (information, bien, argent) entre des comportements.

Relations inter-couches

C'est là que le modèle prend toute sa valeur. Les relations de réalisation connectent les couches verticalement : un Application Service réalise un Business Service. Un Technology Service réalise un Application Service. Un Artifact réalise un Data Object. Un Application Component est assigné à un Node. Ces relations inter-couches permettent la traçabilité de bout en bout, de la stratégie à l'infrastructure.

Les propriétés : enrichir le modèle au-delà du visuel

Un modèle ArchiMate sans propriétés est une cartographie. Un modèle avec des propriétés bien choisies est un référentiel d'architecture exploitable. Archi permet d'ajouter des propriétés personnalisées (clé-valeur) à chaque composant et à chaque relation. C'est un levier sous-estimé.

Quatre propriétés me semblent indispensables dès le départ. La première est le **propriétaire** (*owner*) : qui est responsable de ce composant ? Sans cette information, l'analyse d'impact reste théorique. La deuxième est le **statut du cycle de vie** (*lifecycle status*) : le composant est-il en production, en cours de décommissionnement, prévu, en développement ? On peut utiliser des valeurs comme *current*, *target*, *phase-out*, *retired*. Cette propriété permet de créer des vues différenciées entre l'architecture actuelle et l'architecture cible.

La troisième propriété essentielle est la **criticité métier** : ce composant est-il critique, important ou standard ? Cela oriente les décisions en cas d'arbitrage. La quatrième est un **identifiant de référence** qui relie le composant ArchiMate à un enregistrement dans un autre système : CMDB, référentiel applicatif, outil ITSM. Sans ce lien, le modèle vit en silo.

On peut aller plus loin avec le temps : coût annuel, date de fin de support, niveau de conformité, dette technique estimée. Mais ces quatre propriétés de base suffisent pour passer d'un dessin à un outil de gouvernance. L'important est de fixer des conventions de nommage et de valeurs dès le début, et de s'y tenir.

Les viewpoints : montrer la bonne chose au bon public

ArchiMate propose une vingtaine de viewpoints prédéfinis. Un viewpoint est une configuration de la vue qui filtre les types de composants et de relations autorisés. L'idée est de ne montrer à chaque partie prenante que ce qui la concerne.

En pratique, on en utilise quelques-uns régulièrement. Le **Layered Viewpoint** donne une vue transversale de toutes les couches : c'est la vue « big picture » qu'on montre en comité d'architecture. L'**Application Cooperation Viewpoint** se concentre sur les interactions entre composants applicatifs : utile pour les revues d'intégration. L'**Infrastructure Viewpoint** descend dans la couche Technology pour documenter les nœuds, réseaux et logiciels d'infrastructure. Le **Business Process Cooperation Viewpoint** montre comment les processus métier interagissent entre eux et avec les services applicatifs.

Rien n'empêche de créer des vues sans viewpoint contraint, c'est d'ailleurs ce que je fais le plus souvent au début d'un travail de modélisation. Le viewpoint est un outil de communication, pas une contrainte de modélisation.

Conventions et bonnes pratiques que j'applique

Nommer les composants de manière explicite et cohérente. Un Application Component s'appelle par le nom du système tel qu'il est connu dans l'organisation, pas par un acronyme cryptique. Les services portent un nom qui commence par un verbe ou qui décrit le comportement exposé. Les processus indiquent ce qu'ils produisent.

Documenter le champ *documentation* de chaque composant dans Archi, même avec une phrase. C'est ce texte qui apparaît au survol et dans les rapports. Un composant sans documentation est un composant que personne ne comprendra dans six mois.

Ne pas modéliser ce qu'on ne maintiendra pas. Un modèle obsolète est pire qu'une absence de modèle parce qu'il donne une fausse confiance. Si un périmètre n'est plus activement gouverné, mieux vaut le retirer ou le marquer clairement comme non maintenu.

Utiliser les couleurs avec parcimonie. Archi permet de colorer les composants. Les couleurs par défaut par couche (jaune pour Business, bleu pour Application, vert pour Technology) sont un bon point de départ. Ajouter des couleurs personnalisées uniquement si elles portent une information : rouge pour les composants à décommissionner, par exemple.

Conclusion

ArchiMate n'est pas un outil de plus. C'est un changement de posture : passer du dessin au modèle, de l'image figée au référentiel vivant. La courbe d'apprentissage est raisonnable pour un architecte qui accepte de commencer petit et d'itérer. Le logiciel Archi, gratuit et open source, supprime la barrière financière.

Références

[1] ArchiMate 101 : A Practical Introduction. Traduction française, CC BY-SA 4.0. 21 déc. 2021.

[2] Phil Beauvoir. Archi – Open Source ArchiMate Modelling.

[3] Emmanuel Georjon. Archimate : Présentation. 27 avr. 2021.

[4] The Open Group. ArchiMate 3.2 Specification. 2022.

[5] The Open Group. ArchiMate Overview.

[6] The Open Group. Understanding the Basics – An Introduction to the ArchiMate Modeling Language.