Stratégies de servicisation du SI

Transformation du SI en offre de services pour l'entreprise et ses partenaires

2020 Nov.

NOVEMBRE 2020

Cigref
RÉUSSIR
LE NUMÉRIQUE

Stratégies de servicisation du SI

La transformation du SI en offre de services pour l'entreprise et ses partenaires

Synthèse

Pour devenir plus agiles et innovantes, les grandes entreprises et administrations publiques ont déjà ouvert leurs systèmes d'information (SI) aux équipes en interne ou à leur écosystème proche ou étendu (fournisseurs, clients, vendeurs, partenaires, etc.), avec des services cloud et elles proposent maintenant des services techniques ou fonctionnels, rendus accessibles par le biais d'APIs. Cette ouverture des SI par cette approche de services est aussi appelée servicisation. Ce rapport qui s'adresse à l'ensemble des collaborateurs, étudie comment s'articule la stratégie de servicisation au niveau des SI et plus généralement de l'entreprise.

L'ouverture des SI par cette approche de services répond en effet à plusieurs enjeux aussi bien business qu'IT : homogénéiser les processus métiers et les automatiser ; personnaliser l'expérience client ; coconstruire de nouvelles offres/services avec ses partenaires ; améliorer l'efficacité opérationnelle, réduire le *time to market ;* et enfin donner accès aux applications et services tout en gardant le contrôle du savoir-faire. Les enjeux IT sont de mieux servir les métiers, supporter les processus, offrir une certaine autonomie pour les équipes qui consomment les APIs, et diminuer les coûts.

L'ouverture du SI amène les entreprises à offrir aussi bien aux partenaires internes qu'externes, un portefeuille de services cohérents. Pour cela elles appliquent plusieurs principes clés. Tout d'abord, les entreprises transforment leurs SI pour les rendre modulaires. Ensuite, elles adoptent une démarche de « sécurité *by design* » des APIs pour éviter les principaux risques de compromission. Enfin, les grandes entreprises et organisations publiques segmentent leurs APIs par type ou domaine, tel que le préconise leur cadre d'architecture.

Certaines industries se sont en effet organisées pour déterminer entre partenaires, des standards d'interopérabilité dans un cadre d'architecture. Ce cadre d'architecture définit des spécifications communes d'API afin de standardiser des modèles de données et certaines interactions techniques. Il convient si ce cadre existe d'en faire usage afin d'offrir le meilleur niveau de qualité et d'interopérabilité possible, plutôt que de créer ses propres modèles et API depuis zéro.

Les grandes entreprises construisent les APIs comme des produits à part entière et de façon « *open by design* », tenant compte de l'expérience développeurs. Elles cherchent enfin à maximiser la réutilisation des APIs, démarche facilitée par une solide gouvernance.

Les participants du groupe de travail Cigref préconisent plusieurs bonnes pratiques dans cette démarche de servicisation du SI :

- Favoriser les standards de l'industrie & utiliser un cadre de référence ;
- Définir collaborativement l'interface technique avant de l'implémenter ;
- Définir des règles de design ;
- Soigner la documentation ;
- Soutenir et acculturer les équipes ;
- Faire preuve de pragmatisme ;

- Mettre en place la gouvernance des APIs ;
- S'appuyer sur une solution d'API management.

La prise en compte de ces bonnes pratiques permet d'accélérer les démarches d'agilité des entreprises et d'envisager à terme la mutualisation de certains process métiers ou des développements communs à leur secteur d'activité ou écosystème. Ainsi les entreprises se concentrent sur leur cœur de métier et là où elles apportent de la valeur.

Remerciements

Nos remerciements vont à Françoise Moumen, Responsable d'architecture groupe Orange, et Thierry Souche, CIO du groupe Orange qui ont piloté cette réflexion, avec le support de l'expert technique Vincent Tassy, architecte IT Air France-KLM ainsi qu'à toutes les personnes qui ont participé et contribué à ce groupe de travail Cigref :

Franck ATGIE - ENEDIS
Christophe BANDINI - ESSILOR INTERNATIONAL
Vincent BELROSE - LVMH
Frédéric BERQUEZ - MATMUT
Irinel BETA - GROUPE ADP
Thierry BEY - GROUPE PSA
Nicolas BONAVITA - COVEA
Abdo BOU-MANSOUR - PLASTIC OMNIUM
Philippe BRON - MINISTERE DE L'INTERIEUR
Marie-Laure BUREL - CNAF
Jérôme CECCALDI - AXA
Pierre-Louis CHAMELOT – MINIST. ÉCOLOGIE & TERRITOIRES
Claudio CIMELLI - MINISTÈRE DE L'ÉDUCATION NATIONALE
Renaud CLEAC'H - NAVAL GROUP
Jérôme DE GALLÉ - AXA
Vincent DE PONTHAUD - AXA
Tanguy DEFLANDRE - BANQUE DE FRANCE
Sylvie DEMAREST - ORANGE
Georges DESRAY - COVEA
Philippe DOUBLET - EDENRED
Hassan DRISS - MINISTERE DE L'INTERIEUR
Jean-François DUPITIER - POLE EMPLOI
Marc ESCUYER - BANQUE DE FRANCE
Daniel FLEURENCE - MINISTÈRE DE L'INTERIEUR
Pierre-Xavier FOUILLÉ - NAVAL GROUP
Michel FRECHOU - ENEDIS
Grégory GONZALEZ - MAÏSADOUR
Julien GROSSO - ORANGE
Fabrice GUÉANT - AXA
Pierre GUISERIX - CAISSE DES DÉPÔTS
Jean HUOT - VINCI
Patrick JARJOURA - CNAF
Olivier JAUZE - MICHELIN
Erwan JEGADEN - VINCI
Cédric JUBLOT - EIFFAGE
Emmanuel KURTH - GROUPE EGIS
Raynald LASOTA - FRANCE TELEVISIONS
Thierry LEJEALLE - POLE EMPLOI
Jean-Pierre LOUSTAU - BNP PARIBAS
Sébastien MARIE - MATMUT
Céline MASSY - CEA
Françoise MOUMEN - ORANGE
Mikael PETIT - GRDF
Nicolas REMES - HARMONIE MUTUELLE
Ludovic ROBERT - ORANGE
Jean-Louis ROCCHISANI - SOCIÉTÉ GÉNÉRALE
Erwan SALMON – MINIST. ÉCOLOGIE & TERRITOIRES
Valery SIMON - BANQUE DE FRANCE
Thierry SOUCHE - ORANGE
Vincent TASSY - AIR France-KLM
Olivier TRAN - TOTAL
Jean-Robert VIVENT - VEOLIA

Le Cigref remercie les intervenants lors des différentes réunions de travail :

- Vincent TASSY – Air France-KLM
- Sylvie DEMAREST – Orange
- Ludovic ROBERT – Orange
- Jean-Louis ROCCHISANI – Société Générale
- Olivier JAUZE – Michelin
- Jérôme de GALLÉ – AXA
- Marie-Laure BUREL – CNAF
- Laurent FISCHER – Accenture
- Pierre-Louis CHAMELOT – Ministères Ecologie et Territoires
- Christophe BANDINI – Essilor
- Julien GROSSO – Orange
- Benoît DARDE – Wavestone.

Ce document a été rédigé par Marine de Sury, Directrice de mission Cigref, avec la participation de Françoise Moumen et de Vincent Tassy.

Table des matières

Table des figures

Introduction

Les entreprises cherchent à explorer de nouvelles opportunités face à de multiples formes de disruptions et à la transformation parfois rapide des modèles d'affaires à l'ère digitale. Les entreprises et administrations publiques se transforment pour valoriser leurs données, multiplier les partenariats et réinventer les modèles d'affaires, comme le soulignaient déjà la fondation Cigref et les groupes de travail dans l'esquisse du design de l'entreprise à 2020, synthèse publiée en 2015.

Les grandes entreprises ont pris conscience de la nécessité de remettre le client au centre de leurs préoccupations et accordent autant d'importance à l'expérience client qu'au produit. Elles exploitent la donnée, produite en temps réel ou non, en interne ou au sein de leur écosystème (fournisseurs, clients, vendeurs, partenaires, etc.) pour : favoriser l'innovation commerciale et la personnalisation des services, s'adapter aux usages et à l'évolution des comportements, et enfin répondre à des attentes nouvelles de leurs clients et aux défis des marchés de demain. Mais cette transformation profonde ne s'arrête pas à la proposition de valeur, elle se répercute sur l'ensemble des composantes de l'entreprise : ressources, compétences, flux de revenus et structure de coût, SI (Système d'Information), organisation, opérations, etc. Les SI des entreprises sont de plus en plus complexes et cette complexité constitue parfois un frein aux changements et peut même leur faire manquer ou reporter des opportunités. Pour devenir plus agiles et innovantes, les grandes entreprises et administrations publiques ouvrent leur architecture d'entreprise, leur système d'information (SI), aux équipes internes ou à leur écosystème, avec des APIs[1], des services fonctionnels ou techniques basés sur un cadre global d'architecture. Les entreprises avaient déjà commencé cette ouverture avec les différents services cloud – stockage de données, traitements analytiques, hébergement d'applications, etc. C'est ce qu'on appelle la **servicisation du SI**. Dans ce rapport, le périmètre étudié de la servicisation du SI est celui des APIs au sens large, quels que soient les protocoles utilisés, ou les modes d'interaction (synchrone/asynchrone).

Le groupe de travail Cigref sur les nouvelles stratégies de plateforme avait souligné dans son rapport « Nouvelles stratégies de plateforme »[2], publié en décembre 2019, que l'émergence des modèles d'affaire de type plateforme ne faisait que renforcer ce besoin de transformation des entreprises. Il mentionnait dans sa conclusion l'importance d'étudier la servicisation du SI, nécessaire pour implémenter les stratégies de plateforme de façon efficiente.

L'objectif de ce présent rapport qui s'adresse à l'ensemble des collaborateurs au sein d'une entreprise, est d'étudier comment se traduit la stratégie de servicisation au niveau des SI : les exigences techniques, les nouveaux types de données, la sécurité pour construire un SI performant, et plus globalement l'organisation, les opérations, la gouvernance à mettre en œuvre.

[1] Les APIs sont des interfaces de programmation (*Application Programming Interface*) qui assurent l'accès aux services et données d'un tiers et permettent à des applications externes/internes à l'entreprise de se « brancher » sur une application-ressource pour échanger des données.

[2] https://www.cigref.fr/nouvelles-strategies-de-plateforme-strategie-conception-mises-en-oeuvre

Le rapport précise tout d'abord à quels enjeux la servicisation des SI répond, donne des principes clés d'architecture et de gouvernance et ensuite partage des bonnes pratiques dans les démarches de servicisation du SI, relevés par les participants du groupe de travail (GT) Cigref « Stratégie de servicisation du SI ». Pour ouvrir le SI, il ressort de cette année de travail l'importance de s'appuyer sur un cadre d'architecture (*framework*) existant plutôt que de partir de zéro. Le cadre ODA (*Open Digital Architecture*), utilisé beaucoup dans le secteur des télécommunications mais pas uniquement, présenté à la fin du rapport, permettra au lecteur de comprendre en quoi un *framework* consiste et mieux appréhender l'intérêt d'utiliser un tel cadre d'architecture.

1 Présentation des enjeux Business et SI

Le numérique donne l'opportunité de redéfinir les relations entre l'entreprise et ses clients finaux, entre ses équipes en interne ou encore au sein de son écosystème (fournisseurs, clients, vendeurs, partenaires, etc.).

1.1 Enjeux business ou Métier

La servicisation des SI répond à plusieurs enjeux business : communication globale instantanée, maîtrise des données en temps réel, digitalisation et automatisation de nombreux processus pour améliorer l'expérience client ou l'offre, développement de nouveaux canaux de distribution, rapidité de mise en œuvre, etc. La servicisation facilite également la personnalisation des services. Ces enjeux business se déclinent selon leur finalité :

- **Rendre homogènes les processus métiers et les automatiser** : les processus métiers ne sont pas toujours homogènes au sein des différentes filiales d'une même entreprise. C'est souvent le cas après des fusions/acquisitions, par exemple. Pour autant, les entreprises cherchent à avoir des échanges et communications les plus efficaces. Pour pouvoir livrer son produit ou service au client dans les temps, l'entreprise cherche à avoir des processus les plus automatisés possibles. Cela doit pouvoir s'étendre à tout son écosystème et faciliter les interactions avec des partenaires ou intégrer des solutions de tiers dans son SI le plus facilement possible.
- **Offrir une meilleure expérience utilisateur** : pour l'améliorer, l'interaction avec l'utilisateur doit bien souvent être « multi canal » c'est-à-dire lui permettre de passer du site web à l'application sur *smartphone*, *chatbot*, ou tout autre support, en tirant pleinement profit des capacités de ces équipements, et tout en assurant une continuité dans la chaîne des processus.
  Afin de **personnaliser l'expérience client,** les entreprises souhaitent la digitaliser pour la personnaliser dans le respect des règles de sécurité et de vie privée (recueillir les données clients, les analyser, les exploiter dans les processus opérationnels, les agréger/anonymiser, éventuellement les commercialiser). L'entreprise, forte du consentement client pour utiliser ses données, peut alors collecter les informations requises et servir le client de façon plus fine en utilisant par exemple des techniques d'IA. Cependant, les entreprises se heurtent souvent à la difficulté de devoir gérer une multiplicité de référentiels de personnes ou de services, parfois désynchronisés. L'entreprise cherche donc à organiser son SI de façon à n'avoir qu'une seule représentation de ces ressources clés, favorisant par là même une gestion centralisée et uniformisée. Cette centralisation peut être organisée par domaines métiers, chaque domaine étant responsable des données qui le concerne et garant de son intégrité. Cette approche permet par exemple, au travers d'un parcours en temps réel, de gérer un seul composant pour différents frontaux digitaux (application mobile, extranet…), celui-ci orchestrant l'appel à différents services.
- **Coconstruire de nouvelles offres et de nouveaux services avec ses partenaires et améliorer l'efficacité opérationnelle** : il est important que l'entreprise identifie avec ses partenaires les périmètres communs qui leur permettront d'améliorer ou de diversifier leurs offres. Pour mettre en place ses nouveaux services, l'entreprise est amenée à ouvrir son SI à tout l'écosystème

impliqué. Cela nécessite un cadre et s'inscrit plus globalement dans un projet de transformation au niveau de toute l'entreprise pour intégrer à tous les niveaux cette agilité opérationnelle. L'entreprise peut alors répondre rapidement aux changements du marché (*Time-To-Market)*. Une telle démarche permettra par exemple à une entreprise de pouvoir distribuer divers produits issus de différents producteurs en conservant un unique parcours homogène, tirant profit des APIs des partenaires.

- **Donner l'accès aux applications et services tout en gardant le contrôle du savoir-faire** : la servicisation fait évoluer les SI de façon à pouvoir agréger des outils technologiques variés qui facilitent des développements rapides pour tester des modèles d'affaires innovants ou offrir de nouveaux modèles d'affaires tout en assurant la sécurité et la traçabilité. Les démarches de servicisation permettent de donner accès, de manière coordonnée et efficace, aux applications et services et ainsi répondre à la demande des métiers, des partenaires et plus généralement de l'écosystème tout en gardant le contrôle des SI.

1.2 Enjeux SI

La servicisation du SI répond également à des enjeux internes pour mieux servir le business, rendre plus fluide les processus et diminuer les coûts. En effet, la Direction des Systèmes d'Information (DSI) cherche à augmenter la valeur apportée aux Métiers, diminuer le coût et le temps nécessaires à l'intégration de nouveaux systèmes. Développer une seule fois la même fonction / le même processus et le mettre à disposition dans différents contextes conduit naturellement à une telle optimisation des coûts et des ressources. Certaines entreprises grossissent par acquisitions et se retrouvent avec une « galaxie » de SI hétérogènes. Exposer leurs applications et leurs services au travers d'une plateforme assure l'interaction entre les différents SI mais aussi permet de mutualiser les données dans des référentiels dédiés, évitant alors des réplications, souvent sources d'erreurs.

La DSI doit également organiser la transformation progressive des SI existants vers les nouveaux systèmes. L'utilisation d'APIs normalisées facilite une transition transparente et sans à-coups depuis le SI patrimonial (*legacy*). En effet, cela permet de créer une interface stable qui masque les spécificités d'implémentation sous-jacentes (issues du SI patrimonial ou remplacées progressivement par des composants plus modernes, qu'ils soient hébergés localement ou externalisés vers le cloud. La modernisation de ces composants sous-jacents est une opération complexe qui nécessite de procéder par étapes raisonnables à gérer. En effet, la DSI cherche à transformer son SI tout en assurant la sécurité des données sensibles et le niveau de service du SI. Pour cela, un cadre d'architecture aide à identifier les étapes puis à les piloter. Ces différentes étapes peuvent être par exemple :

- Identifier les fonctions clés et normaliser leur usage ;
- Optimiser l'expérience développeur ;
- Valoriser le SI en étudiant comment il est utilisé ;
- Assurer la résilience du SI et les données sensibles ;
- Ouvrir aux partenaires et à l'écosystème.

Enfin, la servicisation du SI le rend plus modulaire ce qui facilite la mise en œuvre de nouveaux services ou d'applications, et améliore l'efficacité opérationnelle en donnant de l'autonomie aux équipes - développeurs et utilisateurs.

2 Principes clés

Les participants du groupe de travail ont identifié plusieurs principes clés dans la servicisation du SI répondant aux enjeux business et SI, décrits précédemment.

2.1 Rendre les systèmes modulaires

Les entreprises rencontrent plusieurs écueils :

- Les données en silo offrent une vue fragmentée ;
- L'architecture des SI est bien souvent monolithique ;
- Les changements ont un fort impact dans les SI qui sont de plus en plus complexes.

Une approche classique consiste à passer d'une architecture monolithique à une architecture d'applications web basée sur des APIs. Autrement dit, on isole **dans le SI tous les systèmes *legacy* en interaction avec le client** sur les différents canaux possibles pour offrir une expérience client sans contrainte. Un cadre d'architecture favorisera pour cela par exemple, l'isolation des systèmes de gestion orientés clients, des systèmes de gestion orientés catalogue ou offres et des systèmes de gestions fournissant les solutions techniques. Si la plupart des interactions demeurent synchrones avec une forte dépendance entre les couches du SI, on peut noter une démocratisation croissante d'architectures asynchrones aussi appelées évènementielles (*Event Driven Architecture*) qui permettent une meilleure isolation des applications entrant en jeu dans les processus. Enfin, l'avènement de nouveaux styles d'architectures telles que les architectures microservices permettent d'aller encore plus loin dans la modularité, favorisant, et ce de manière standardisée, l'indépendance des services mettant en œuvre ces interactions asynchrones.

Les participants du groupe de travail Cigref préconisent au niveau de l'architecture d'entreprise d'isoler les processus des IHMs (Interface Homme Machine) notamment pour construire des solutions permettant nativement la multicanalité : le découplage du *FrontEnd* et du B*ackEnd*[3] permet de revoir la présentation d'un portail sans impact sur les processus des *BackEnds*. Ainsi l'offre client a la possibilité d'évoluer sur un nouveau canal - par exemple le canal mobile ou un site web renouvelé - avec de nouveaux services et offres, sans pour autant modifier les processus sous-jacents ni altérer la cohérence de l'expérience utilisateur. Cette séparation permet alors de les gérer de manière plus cohérente, indépendamment des référentiels et des IHMs.

Les participants membres du GT Cigref préconisent également de dissocier la notion de services de la notion d'APIs : il s'agit de définir des services métiers fonctionnels qui couvrent un niveau de processus métier (fonctions et données) et qui interagissent entre eux via des APIs. En effet, chaque service peut disposer de différentes APIs suivant le cas d'usage : par exemple usage externe public versus en interne.

[3] En informatique, un *BackEnd* (parfois aussi appelé un arrière-plan) est un terme désignant un étage de sortie d'un logiciel devant produire un résultat. On l'oppose au *FrontEnd* (aussi appelé un frontal) qui lui est la partie visible de l'iceberg.

Afin de limiter les réplications de données au sein d'un domaine métier ou entre 2 domaines, les participants recommandent de déterminer le domaine responsable pour chaque type de données. L'accès aux données se fait alors au travers d'APIs auprès du domaine responsable de les tenir à jour. Cela facilite de plus la mise en œuvre des procédures liées au respect de la GDPR.

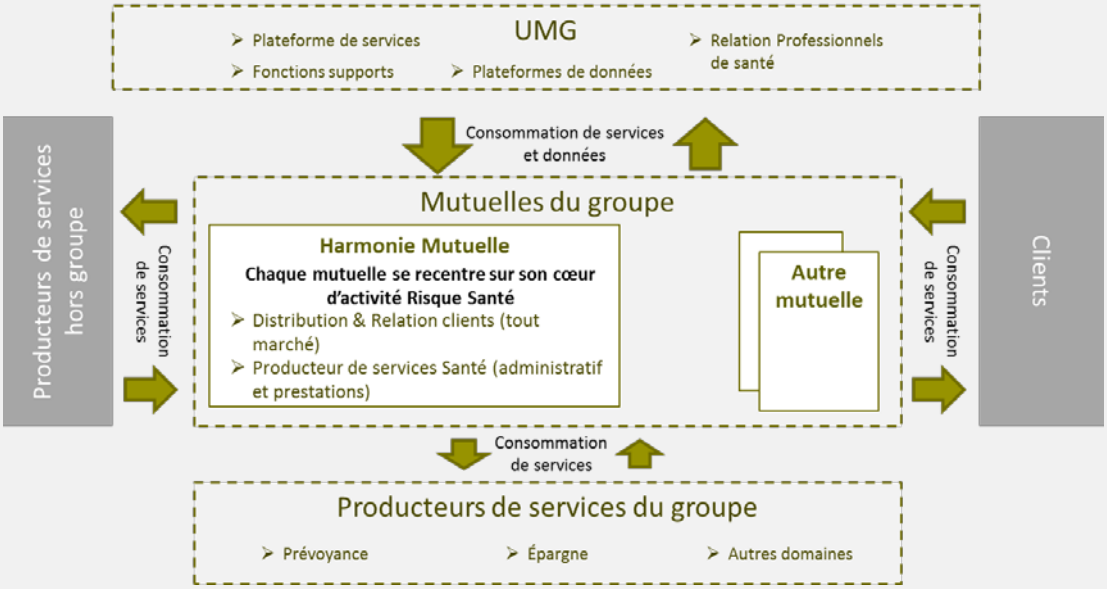
Pour mettre en œuvre ce type d'approche de manière pragmatique, des étapes intermédiaires ou la mise en œuvre de systèmes de virtualisation/fédération de la donnée peuvent être envisagées en fonction de l'existant et de la complexité du SI.

Un autre niveau d'isolation s'effectue dans l'orchestration au sein même du SI. Il s'agit de découpler les processus des couches basses, à savoir des solutions technologiques choisies. Ainsi, changer de fournisseur pour des briques applicatives n'a ainsi pas d'incidence sur les processus et orchestrations à condition bien sûr que ces nouvelles « briques applicatives » aient bien exactement la même couverture de fonctions et offrent bien les APIs attendues par la couche process.

Un dernier point concerne l'importance de la modularité des SI pour réduire le *time to market,* c'est-à-dire le délai de mise sur le marché qui inclut le temps nécessaire pour le développement et la mise au point d'un projet ou d'un produit, avant qu'il puisse être lancé sur le marché. Les entreprises n'hésitent pas à s'associer à des partenaires au sein de leur écosystème, et même à l'extérieur, pour développer de nouveaux modèles d'affaires. C'est pourquoi, les participants préconisent de **penser et définir une architecture ouverte**, au-delà de son propre cœur de métier. Accenture propose par exemple de construire au sein du SI, une entité avec une organisation propre pour permettre de tester rapidement de nouvelles initiatives métier en combinant les APIs et services de l'entreprise et ceux de l'écosystème extérieur dans un environnement de type « *plateform as a service* » permettant ainsi un passage rapide du prototype à l'échelle et donc stimuler l'innovation des équipes produit.

A titre d'exemple, au sein du groupe VYV, Harmonie Mutuelle joue un rôle de producteur et distributeur de produits d'assurance santé et un rôle de distributeur de produits d'assurance fournis par d'autre entités du Groupe et/ou par des partenaires.

Dans ce modèle, la relation clients modulaire et multicanal est portée par le distributeur



Au niveau de la gouvernance du SI, cela se traduit par une :

- Interopérabilité :
  - Ouverture des SI
  - Ouverture et maîtrise de leurs données
- Généralisation du temps réel dans leurs échanges
- Maîtrise de la sécurité des données et des évolutions règlementaires (RGPD, gestion du consentement)
- Remontée d'indicateurs assurant le respect des engagements
- Gestion de l'asynchronisme des processus au travers du *Case Management*

Nicolas Remes, Harmonie Mutuelle Groupe VYV

2.2 Mettre en place une démarche de *Security by design*

La sécurité a, par le passé, trop longtemps été considérée dans les projets IT comme un élément à traiter à la fin des développements, comme d'autres aspects non fonctionnels (haute disponibilité, monitoring, etc). Or la sécurité doit être adressée dès les premières phases du projet, depuis le design, pendant le développement, les tests et jusqu'au suivi en production. Ceci est vrai pour le développement applicatif en général mais devient encore plus critique dans le développement et l'exposition d'APIs. Lors de la conférence RSA 2019, Akamai indiquait qu'en 2018, le trafic lié aux APIs représentait 83% des échanges sur le web ! Cette augmentation significative de la surface d'échanges et donc de la surface d'attaques en fait dorénavant la cible privilégiée des hackers. De plus, les bonnes pratiques en termes de sécurisation des APIs au moment du développement ne sont pas encore suffisamment répandues.

OWASP[4] (The Open Web Application Security Project®) qui se concentrait jusqu'à présent sur la sécurité des applications web publie à présent son Top 10 sur la sécurité des APIs[5] pour aider les concepteurs à éviter les principaux risques de compromission d'une API.

Si l'agilité a été mise en œuvre, une approche DevSecOps permet d'intégrer élégamment la gestion de la sécurité au cycle de vie des APIs facilitant ainsi l'application des meilleures pratiques depuis la phase de design, jusqu'à la mise en production et l'exploitation en passant par le développement et le test, le tout sans perturber l'efficacité des développeurs. Certaines sociétés se sont spécialisées sur l'outillage de ces phases liées à la sécurité et permettent de facilement se faire une idée des principaux risques de certaines APIs.

L'Internet de objets (IoT) représentait 2/3 du trafic API enregistré en 2018 - chiffres donnés par Akamai en 2019 lors de la conférence RSA. Ces objets qui se multiplient de façon exponentielle chaque année, ont parfois des capacités limitées en termes de *processing*, ce qui limite leur possibilité de sécurisation. Ils sont donc une source de trafic potentiellement dangereux qu'il convient d'évaluer, de surveiller

[4] https://owasp.org/www-project-api-security/
[5] https://github.com/OWASP/API-Security/raw/master/2019/en/dist/owasp-api-security-top-10.pdf

et/ou de gérer sur des plateformes dédiées, hors du SI quand c'est possible. Cependant, de nombreux progrès ont été faits depuis pour rendre ces équipements plus sûrs.

2.3 Concernant les APIs

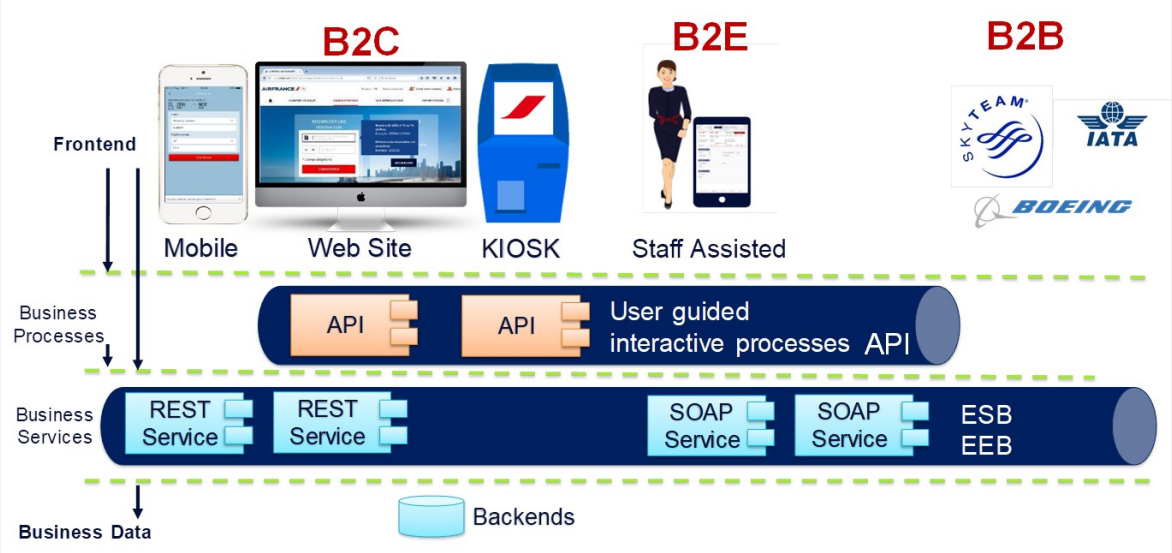
Les APIs sont développées pour assurer l'interconnexion et les échanges entre les services et données d'une entité ou avec un tiers, qu'il soit interne ou externe à l'entreprise. Elles doivent être architecturées de façon rigoureuse et assurer une bonne granularité.

2.3.1 Différencier les types d'APIs

L'utilisation d'un cadre d'architecture amène généralement à organiser les APIs en plusieurs catégories telles que :

- Les **APIs de données** (ou *Core* API ou *system* API ou *resources* API) qui permettent d'uniformiser l'accès aux données transactionnelles ou aux référentiels de l'entreprise tout en masquant les spécificités techniques induites par les *BackEnds* (éventuellement issus du SI patrimonial).
- Les **APIs de process** permettent d'orchestrer les appels aux divers systèmes intervenant dans la réalisation des processus de l'entreprise quand ceux-ci sont entièrement automatisés (exemple : besoin de créer un client pendant une prise de commande) et coordonnant les acteurs humains dans le cas de processus partiellement automatisés. Ces APIs assurent une cohérence du point de vue de l'utilisateur final quelles que soient les IHM (Interfaces Homme-Machine) qui y font appel.
- Les **APIs de présentation** (ou *Experience* API). A contrario des deux premières catégories, celles-ci ne sont mises en œuvre que pour répondre à des besoins spécifiques de distribution des données pour un canal digital particulier (par exemple, exposer les données avec GraphQL pour optimiser le chargement depuis une application mobile, limiter le contenu exposé pour une API grand public comme OpenAPI). Les développements *Mobile First*, qui peuvent être sujets à une latence ressentie par l'utilisateur, peuvent tirer profit de ces couches d'API afin de pouvoir précharger du contenu et optimiser celui-ci spécifiquement pour cet usage.

> A titre d'exemple, le groupe Air France–KLM a fait le choix d'organiser en deux couches distinctes des services technico-fonctionnels complètement autonomes et capables de réaliser de manière complètement automatisée une fonction métier.
>
> Placée au-dessus de ces services/fonctions, se trouve une couche d'APIs conversationnelles permettant de factoriser les processus clés d'interaction avec les utilisateurs et de les rendre disponibles au travers de différents canaux digitaux.



Vincent Tassy, Air France-KLM

Nous nous focalisons dans ce rapport davantage sur les deux premiers types d'APIs qui sont les couches incontournables pour mettre en œuvre une APIsation efficace du SI ; la troisième étant optionnelle, parfois mise en avant dans la méthodologie de certains fournisseurs d'infrastructure et qui peut être réalisée par d'autres moyens que les APIs.

Les APIs de données comprennent les :

- APIs qui facilitent la collecte et la mise à disposition de la donnée ;
- APIs qui permettent de construire des data lakes6 (pour l'industrie, les services, etc.) avec des données nettoyées et intégrées ;
- APIs de gestion des données, c'est-à-dire l'API propre à la réalisation du processus métier. Ces APIs consultent et mettent à jour les données. Pour favoriser l'innovation, les opérations de consultation peuvent être rendues accessibles à tous les composants du SI sous réserve d'une bonne gestion des droits.

Les APIs de type process incluent les :

- APIs propres à la gestion du processus métier qui fonctionnent en question/réponse avec le *FrontEnd* (données saisies permettant les mises à jour de données). Synchrones ou asynchrones, elles sont utilisées pour les interactions entre *FrontEnd* et *BackEnd*, ou entre *BackEnds*. Ces APIs fournissent les données attendues par les acteurs du processus afin que ceux-ci puissent réaliser les tâches qui leur incombent (tâches manuelles).
- APIs de souscription et publication d'évènements qui permettent d'interagir de manière asynchrone par notification sur modification de l'état d'un objet métier.

[6] Concept lié à la mouvance du big data, le lac de données (ou *data lake* en anglais) désigne un espace de stockage global des informations présentes au sein d'une organisation. Il s'agit de le faire avec suffisamment de flexibilité pour interagir avec les données, qu'elles soient brutes ou très raffinées.

Il est à noter que la cohabitation entre APIs synchrones et APIs asynchrones mérite un point d'attention particulier pour ne pas exposer l'utilisateur final à des erreurs de type *TimeOut*. Fort heureusement, les standards évoluent avec par exemple le support d'API REST Asynchrones dans le standard *OpenAPI_Specification* ou encore la création du standard AsyncAPI qui permettent de mieux gérer ces types d'interactions, et ce de manière standardisée. L'adoption de ces types d'interaction asynchrone permet en outre de protéger l'opérabilité du SI en évitant par exemple la propagation des contraintes de qualité de service de couche en couche.

Certaines APIs sont spécifiques à des cadres d'architecture comme l'illustre l'exemple du cadre ODA donné en fin de ce rapport.

2.3.2 Construire l'API comme un produit à part entière

Les APIs constituent un moyen d'intégration et d'ouverture des systèmes dans les SI. Le découpage du SI avec des APIs de données et des APIs process offre la modularité nécessaire pour faire évoluer l'architecture du SI et transformer progressivement le *legacy*. Les coûts IT diminuent avec la réutilisation des applications au travers des APIs, avec la réduction du nombre d'interfaces à destination des partenaires externes et internes.

En plus de faciliter l'évolution des SI patrimoniaux, exposer et partager des APIs est un moyen de créer de l'interaction en interne ou avec des partenaires pour développer de nouveaux modèles d'affaires. Les APIs évoluent alors vers un modèle de plateforme qui interagit avec un écosystème de partenaires pour offrir différents modèles Métier.

Du fait de l'importance que prennent les APIs dans le support des modèles métiers et dans la maintenabilité du SI, il convient de les traiter comme des produits à part entière et non comme des sous-fonctionnalités des applications sous-jacentes au risque de ne pouvoir gérer indépendamment leurs cycles de vie et de ne pas être en mesure d'adapter suffisamment rapidement ces APIs aux changements du marché et des écosystèmes. De plus, comme tout autre produit ou offre de service, ces APIs doivent être dûment documentées et maintenues ! Le GT recommande de gérer les APIs comme des produits, par un *product owner*[7]. Cependant, il n'est pas toujours facile de trouver la bonne personne pour être un bon *product owner* car les APIs sont souvent vues comme des produits techniques.

L'entreprise et plus généralement l'écosystème doivent **construire la stratégie des APIs** qui inclut l'identification, la priorisation, la mise en œuvre, la promotion, l'évaluation et éventuellement la monétisation. La stratégie des APIs a un impact sur l'entreprise et les modèles d'affaires qui se développeront autour.

2.3.3 Designer les APIs pour les développeurs

Favoriser l'expérience développeurs

Beaucoup d'efforts sont faits de nos jours pour soigner les interfaces graphiques des applications afin de les rendre intuitives pour l'utilisateur final. On parle d'expérience utilisateur (UX). Bien qu'étant des

[7] Le profil « *product owner* » est décrit dans la Nomenclature RH des métiers du SI Cigref.

composants techniques, il convient de *designer* avec tout autant de soin les interfaces des services, les APIs, afin d'en rendre l'utilisation intuitive pour les développeurs qui vont en faire usage. On parle d'expérience Développeur (DX).

Définir des APIs *Open by design*

Nombre d'initiatives de servicisation sont lancées pour répondre dans un premier temps à des besoins internes. Dans ce cadre, il peut être tentant de prendre des raccourcis sur le design des services et APIs sachant que producteur et consommateurs appartiennent à la même organisation.

Dans un modèle où les APIs sont un vecteur de création de valeur et donc de revenus, leur utilisation par des personnes étrangères à l'entreprise doit être des plus intuitives. Il faut les concevoir dans cet esprit et non dans l'optique d'un usage purement interne par des personnes qui connaissent parfaitement le SI, le modèle de donnée de l'industrie ou encore les spécificités d'usage propre à l'entreprise. Le temps investi au moment de la conception des APIs est largement compensé par l'assistance qu'il n'est plus nécessaire d'apporter pour aider les partenaires ou les développeurs internes à les mettre en œuvre.

Maximiser la réutilisation des APIs

Les APIs doivent être conçues avec pour but d'être utilisées et réutilisées. Cela nécessite de définir certains processus et une discipline associée au niveau du design et de la mise en œuvre. Ensuite il est important que les développeurs aient le réflexe de réutiliser dès que possible les APIs ou briques déjà développées. Cela nécessite aussi d'avoir la compétence de concevoir et produire des APIs faciles à utiliser dans les SI (documentation pédagogique et simple qui assure l'autonomie des utilisateurs sur l'API et ses attributs) mais aussi au niveau des métiers. Cette réutilisation de processus standards et de services communs offre aux développeurs la possibilité de se focaliser sur les éléments différenciants de l'offre tout en assurant une cohérence des règles métier et en réduisant les coûts de développement.

Pour être réutilisées, les APIs doivent pouvoir être visibles et découvertes. Il est également important de designer des APIs extensibles par design. Cela signifie que dès la conception de l'API on prévoit comment cette dernière pourra être étendue (ajout d'attributs, manipulation de nouvelles entités). A titre d'exemple c'est un travail effectué au TM Forum[8] pour permettre d'étendre les APIs à d'autres contextes d'utilisation. Il faut trouver le bon équilibre entre une API générique et une API désignée pour un besoin. Si elle est trop générique, elle est alors peu utilisable. Si elle est trop spécifique, les APIs seront très nombreuses.

Des entreprises mettent en place un catalogue d'APIs global, réutilisable et pertinent en interne et en externe. Cela nécessite cependant de déterminer avec rigueur des recommandations pour en définir la structure, en adoptant par exemple une organisation par domaine si elle est compréhensible par le plus grand nombre. Pour faciliter cela, les participants du GT recommandent de mettre en place une gouvernance autour des APIs qui est un des principes clés développé ci-après.

[8] TM Forum est une association regroupant plus de 90 000 personnes membres et plus de 850 entreprises membres dont les secteurs d'activité sont en général les télécommunications, les fournisseurs de services comme SAP et Salesforce, les fournisseurs de services cloud, les intégrateurs.

3 Bonnes pratiques dans les démarches de servicisation du SI

3.1 Favoriser les standards de l'industrie et utiliser un cadre de référence

Soigner le *design* des interfaces de service rend l'usage du service plus aisé par les parties prenantes. Certaines industries se sont structurées pour définir des standards d'interopérabilité entre leurs acteurs, définissant des spécifications communes d'APIs afin de standardiser des modèles de données et certaines interactions techniques. Il convient dans ce cas d'en faire usage afin d'offrir le meilleur niveau de qualité et d'interopérabilité possible, plutôt que de créer ses propres modèles et API depuis zéro.

Le standard EDIWheel est porté par l'association EDIWheel dont les membres sont des fabricants de pneus. Ce standard d'échange de données en B2B remplace les échanges physiques de documents entre entreprises (commandes, factures, bons de livraison, etc.) par des échanges, selon un format standardisé, entre ordinateurs connectés par liaisons spécialisées ou par un réseau. Cette standardisation a pour objectif de diminuer la prise de commande par téléphone et portail web pour les remplacer par des appels de système à système.

Michelin utilise le standard EDIWheel afin de soutenir et encourager d'une part l'adoption des APIs en interne et au sein de leur écosystème, et d'autre part la mise en place d'une démarche de standardisation et de normalisation.

Avec la norme EDIWheel, Michelin souhaite également diversifier leurs types de client. La digitalisation permet à Michelin de supporter plus de modèles business pour toucher le consommateur de façon plus directe.

Olivier Jauze, Michelin

L'utilisation d'un standard ouvert permet de s'affranchir de solutions propriétaires et ainsi d'éviter de se trouver dans une situation de dépendance d'un éditeur unique. C'est pourquoi, il est vivement conseillé de se doter d'un cadre de référence ou *framework* en privilégiant le cadre standard utilisé par l'écosystème ou le domaine d'activité afin de capitaliser sur ce qui existe déjà et de définir ce qui manque de cohérence avec ce cadre. Une fois le choix effectué, il faut favoriser et encourager son utilisation en acculturant les collaborateurs au cadre d'architecture et s'assurer que l'APIsation devient un réflexe permettant un développement qualitatif pour pouvoir passer à l'échelle.

Orange utilise le cadre d'architecture ODA défini par l'association de normalisation TM Forum. Cette association regroupe plus de 90 000 personnes membres et plus de 850 entreprises membres dont les secteurs d'activité sont en général les télécommunications, les fournisseurs de services comme SAP et Salesforce, les fournisseurs de services cloud, les intégrateurs. Le cadre ODA est présenté dans la partie suivante de ce rapport.

D'autres cadres existent comme BIAN cadre d'architecture dans le secteur bancaire.

> L'association BIAN (*Banking Industry Architecture Network*) poursuit l'objectif de devenir un standard pour décrire les capacités métiers des banques. L'association comprend des institutions financières, des sociétés informatiques et des académies. L'association a défini un cadre comprenant plus de 300 capacités métiers bancaires qui s'est enrichi d'un portail pour susciter l'innovation et la collaboration avec plus de 140 APIs.
>
> La Société Générale a utilisé ce cadre, en plus d'autres sources internes, pour définir son propre cadre métier commun d'architecture transversal à l'ensemble de ses organisations (banque d'investissement, banque de détail, assurances …). L'ensemble des APIs du groupe sont ainsi classifiées au sein d'un catalogue dans le but de faciliter le partage et de promouvoir la réutilisation des services au sein de l'organisation, d'en favoriser leur gouvernance.
>
> **Jean-Louis Rocchisani, Société Générale**

Dans le domaine de l'aérien, IATA œuvre pour la standardisation des APIs autour de son programme OpenAir, développé dans le cadre de son *Architecture and Technology Strategy Board* auquel AirFrance-KLM participe. Cette initiative tire profit d'un *Airline Industry Data Model* afin de normaliser les échanges et introduit un programme de certification afin de mettre en avant et promouvoir les solutions et partenaires ayant adopté ces bonnes pratiques.

Après avoir standardisé nombre de ses interfaces avec, entre autres, les standards tels que IHE (*Integrating the Healthcare Enterprise*), le monde de la santé est allé plus loin en certifiant les acteurs qui supportent ces standards au travers de campagnes annuelles de tests (IHE Connectathons). Ces certifications apportent aux utilisateurs des solutions compatibles et la garantie d'une réelle interopérabilité entre fournisseurs.

3.2 Définir collaborativement l'interface technique avant de l'implémenter

Il s'agit de comprendre le besoin des utilisateurs finaux et de définir avec eux le cahier des charges avant de développer l'API. Cette bonne pratique souligne encore l'importance du *design* des interfaces de services (API). Ces interfaces agissent comme un contrat technique entre fournisseur et consommateur. Il est donc primordial que les deux participent au design de cette interface. C'est ce qu'on appelle le « *contract first* ». Une entreprise pourra ainsi coconstruire l'interface de service pour qu'elle réponde au mieux au besoin. Là encore, l'utilisation de standards ouverts tels que Open API facilite les échanges même au moment du design.

On peut la rapprocher des pratiques de « *mockup* » (maquette) utilisées pour créer les interfaces graphiques applicatives en impliquant les utilisateurs finaux. On commence par s'entendre sur le *look* et le comportement de l'interface graphique avant de lancer une armée de développeurs dans sa réalisation. Il en va de même pour les APIs pour lesquelles, il est nécessaire de définir leur interface avant de se lancer dans leur implémentation.

3.3 Définir des règles de design

Il existe mille et une manières pour décrire une seule et même API, tant au niveau du modèle de données que de la manière d'invoquer des opérations en passant sur la gestion des erreurs. Ainsi, afin de garantir un bon niveau de cohérence dans un portefeuille de services/APIs en croissance, il convient d'établir des règles de design. Cela est d'autant plus critique que le nombre d'équipes développant les APIs augmente ou qu'il est fait appel à la sous-traitance. La qualité des APIs est clé pour qu'elles soient réutilisées, pour **s'ouvrir vers des modèles de plateforme et pouvoir passer à l'échelle**, notamment sur les APIs ouvertes à l'externe**.**

Plusieurs entreprises ont rendu publiques leurs *API Design Guidelines* qui représentent une excellente source d'inspiration :

Adidas	https://adidas.gitbook.io/api-guidelines/
Zalando	https://opensource.zalando.com/restful-api-guidelines/
Atlassian	https://developer.atlassian.com/server/framework/atlassian-sdk/atlassian-rest-api-design-guidelines-version-1/
Microsoft	https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design
Google	https://google.aip.dev/
Paypal	https://github.com/paypal/api-standards/blob/master/api-style-guide.md
Octo	https://blog.octo.com/designer-une-api-rest/
La Maison Blanche	https://github.com/WhiteHouse/api-standards

3.4 Soigner la documentation

Tout comme n'importe quel produit, les APIs/services doivent être accompagnés d'une solide documentation afin d'en garantir l'adoption. Cette documentation doit couvrir deux aspects :

- Un aspect métier, permettant de comprendre la teneur du service réalisé, la nature des données manipulées et les éventuels comportements attendus. En effet, le service ou l'API est décrit au niveau fonctionnel métier pour définir son rôle, la requête métier ainsi que le déclencheur qui produit la propagation de l'événement et les sous-jacents qui en découlent. A la lecture de ce document de description du service, le Métier comprend ce qu'il offre et les règles et les politiques d'utilisation (confidentialité des données par ex.).
- Un aspect technique destiné aux développeurs qui vont être en charge de les utiliser couvrant notamment les types de données utilisés, les interfaces, les paramètres ou la politique de sécurité, les règles d'usage, les contraintes opérationnelles, la gestion des erreurs techniques, etc. La documentation d'API doit intégrer des exemples en mode bouchon (OK comme KO et ses différents codes retours). OpenAPI est un des standards importants, utilisé dans la documentation technique des API.

3.5 Soutenir et acculturer les équipes

Un ingrédient déterminant de réussite de la servicisation du SI est **l'acculturation des collaborateurs** aux bonnes pratiques, à **l'ouverture** vers les autres équipes et à la **transparence**. La transparence intervient à deux niveaux : **partager** les développements réalisés en interne ainsi que les APIs, et **avoir le réflexe de regarder ce qui existe déjà**. Les équipes doivent être convaincues de l'importance de suivre la règle de design des APIs. Cela nécessite d'acculturer les équipes pour standardiser les développements, préciser les termes et les aspects techniques. Pour transmettre et véhiculer les bonnes pratiques, donner l'exemple avec des champions est un bon moyen. Les participants du groupe de travail Cigref « stratégie de servicisation du SI » préconisent d'acculturer les collaborateurs à l'approche « *API as a product* » plutôt qu'avoir l'approche de livraison de projet, favorisant ainsi le développement itératif des APIs qui s'accorde parfaitement aux pratiques actuelles d'agilité.

Les équipes doivent également prendre en compte les contraintes de sécurité et s'assurer que les APIs sont conçues « *open by design* ». Ces différents points ne sont pas à sous-estimer car ils assurent l'interopérabilité.

La complexité de l'organisation et des systèmes d'information est souvent sous-estimée alors qu'elle impacte directement la servicisation. Ensuite, le manque de langage métier commun et le manque de gouvernance transverse compliquent la servicisation car les synergies sont peu exploitées.

Le soutien aux équipes peut être organisé autour d'une équipe centrale, spécialisée dans le domaine des APIs et dédiée au support technique, à l'éducation voire à la sous-traitance interne des développements pour les équipes n'ayant pas atteint encore la maturité nécessaire. On parle souvent d'API *Factory* pour qualifier ces équipes. Ainsi, plusieurs modèles de « *delivery* » peuvent coexister au sein de l'entreprise avec des développements centralisés ou distribués en fonction de l'évolution des compétences des équipes. Si la taille de l'équipe le permet, l'entité experte centrale peut détacher ses

experts pour travailler directement dans les équipes projet et ainsi favoriser l'acculturation et la montée en compétence. Il n'est pas rare de voir aussi ces entités expertes prendre un rôle « certificateur » pour valider la maturité des équipes sur les développements d'APIs et ainsi se porter garant de leur capacité à développer en toute autonomie, dans le respect des principes édictés dans l'entreprise.

3.6 Faire preuve de pragmatisme

Le pragmatisme est la constante dans la mise en œuvre des démarches. Cela se traduit par une capitalisation au maximum sur l'existant, les standards et sur les cadres d'architectures. Il faut dès la conception penser à l'industrialisation et au déploiement à l'échelle.

Bien que la gouvernance ou le respect des standards soient primordiaux pour la gestion à long terme du portefeuille de services, il faut être en mesure de gérer les aléas ou les contraintes des métiers sans pour autant bloquer les processus et ralentir les projets porteurs de valeur. Ainsi lorsque des exceptions surviennent, il faut les gérer avec pragmatisme et sans intransigeance, sans pour autant les laisser tomber dans l'oubli. Ces exceptions et autres dérogations sont autant d'occasions d'améliorer le processus de gouvernance, mais elles doivent également donner lieu à une traçabilité et une gestion appropriée de la dette technique.

3.7 Mettre en place la gouvernance des APIs

La mise en place d'une gouvernance d'APIs au niveau de l'entreprise avec un outillage adapté permet tout au long du cycle de vie des APIs de pointer les responsabilités lors des différentes actions à mener au niveau des phases stratégiques, de design, de construction et de consommation.

La gouvernance répond à plusieurs objectifs. Tout d'abord, assurer la qualité des APIs en facilitant l'interopérabilité des interfaces et la réutilisation des services pour éviter les doublons. La connaissance du portefeuille de services qui doit être facilement accessible, a vocation à s'améliorer pour répondre aux besoins. Pour cela, suivre l'évolution de la consommation des APIs permet de s'assurer de l'adéquation avec les besoins des consommateurs et d'affiner la stratégie. Ensuite, la gouvernance participe à développer la performance du SI en maîtrisant le parc applicatif et ses flux, en optimisant les solutions avec cohérence et en validant la qualité de l'architecture. La standardisation des processus permet de maîtriser la complexité. Enfin, la gouvernance valide l'alignement du business avec la stratégie d'APIs de la DSI et facilite la priorisation des services à implémenter. En plus d'optimiser et rationnaliser l'investissement, la gouvernance offre des arguments pour financer les services non portés par les Métiers.

Les participants du GT « Servicisation du SI » ont listé plusieurs bonnes pratiques et facteurs de succès dans la mise en place de la gouvernance des APIs :

- Au niveau de l'organisation, les participants recommandent que la gouvernance soit assurée par une équipe pluridisciplinaire qui fasse preuve de pragmatisme. Ils proposent aussi d'impliquer les consommateurs dans le design, d'impliquer les métiers pour la légitimé et pour avoir une expertise forte sur le fond.

- Un autre facteur clé de succès consiste à appliquer les démarches agiles pour les projets de servicisation en lien avec les métiers, à automatiser les processus de gouvernance et enfin à rendre autonome les équipes sur le cycle de vie des services. Maximiser le self-service évite de freiner le processus de mise en œuvre de nouveaux services et applications.
- La standardisation est également recommandée au niveau de la documentation, du catalogue de services, au niveau des données et des services (verbes, comportements, gestion des anomalies), au niveau de la sécurité et la protection des données. La standardisation se traduit par l'acculturation des acteurs de la DSI et des métiers, par la publication de bonnes pratiques autour des APIs. Le standard est parfois amené à évoluer ce qui implique une communication à grande échelle.
- Enfin plusieurs usages sont ressortis comme bonnes pratiques dans la gouvernance des APIs : faciliter l'enrôlement des consommateurs, mesurer les usages des solutions proposées par la servicisation du SI, installer la pratique d'*API as a product*, avoir une gouvernance des données.

Il convient aussi que l'accès à l'API soit facilement sécurisé par le biais de mode d'authentification qui favorise le passage à l'échelle (OAuth, OpenId Connect vs certificat client).

Plusieurs entreprises préconisent la règle de gouvernance suivante : stocker un type de données à un même endroit en nommant un seul responsable. Ensuite, **un service n'est la propriété que d'un seul domaine métier** et d'un seul et unique propriétaire qui gère son cycle de vie au niveau de la stratégie, design, production et consommation. Le fournisseur du service connaît d'autres consommateurs éventuels et il lui incombe de définir le niveau de granularité optimal pour être utilisable au sein du même domaine ou à l'extérieur de son domaine.

Les entreprises qui ont partagé leurs pratiques de gouvernance mises en place en leur sein, précisent que chaque service ou API est décrit à la fois par le métier au niveau fonctionnel et par la DSI qui détermine le *pattern* (patron) d'utilisation et l'aspect technique. La façon dont le service ou l'API est orchestré, est également spécifiée.

Nommer un *Community Manager* autour des APIs permet d'en assurer la promotion, en interne comme en externe, de responsabiliser la gestion des accès, de maximiser l'utilisation et d'en assurer le suivi - par exemple leur taux d'usage - à l'aide d'outils.

Des KPIs (indicateurs de performances) sur les APIs par domaines métier sont mis en place pour évaluer la qualité des services - leur utilisation, leur évolution avec le nombre de versions d'un service opéré, leur taux de réutilisation - et peuvent être envoyés aux architectes du domaine et aux responsables Business IT.

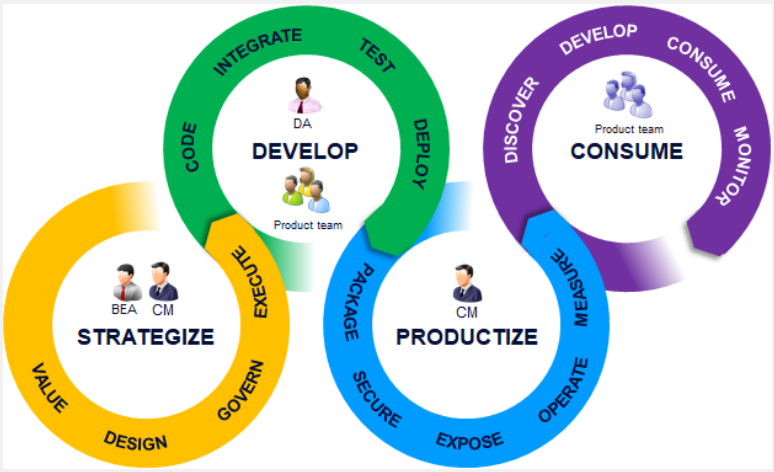
La gouvernance des APIs ne doit pas être sacrifiée pour l'agilité. C'est pourquoi, mettre en place une gouvernance avec un maximum d'étapes automatisées est un objectif à se donner dès le début de sa définition.

Fort d'un usage à l'échelle de la servicisation de son SI depuis plus de quinze ans, le groupe AirFrance–KLM opère un modèle de gouvernance hybride pour ses services et APIs, porté en central par les équipes d'Architecture d'Entreprise du *CIO Office*, en charge de la définition de la gouvernance et des standards de l'entreprise, et s'appuyant sur d'autres architectes, au plus près des métiers et des projets pour sa mise en œuvre.

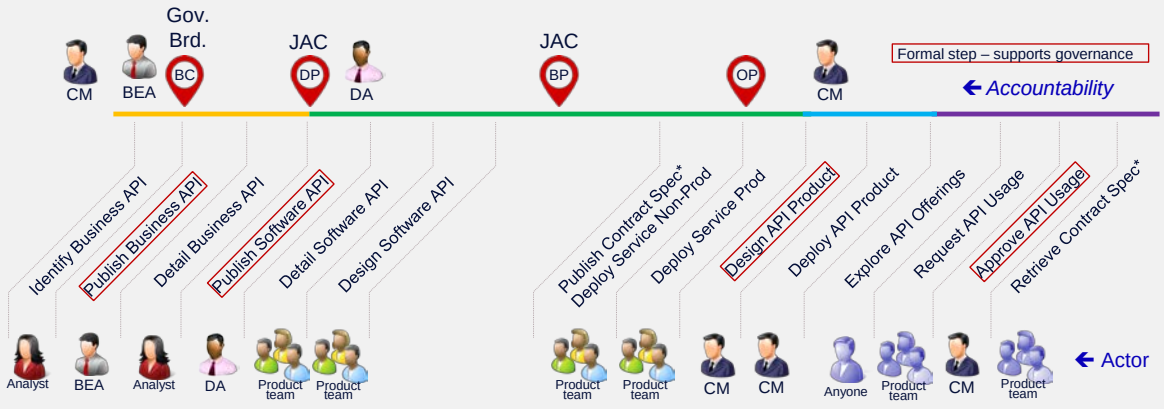
Ainsi, services et APIs sont avant toute chose, stratégiquement définis dans un cadre purement métier par les Architectes d'Entreprise Métiers (BEA), aidés par les Analystes Métiers (BA).

Ils sont ensuite déclinés sous leur forme technique (API, REST, SOAP, Synchrone, Asynchrone, etc.) par les Architectes des Domaines Métiers (DA) avant d'être mis en œuvre par les Equipes Produits.

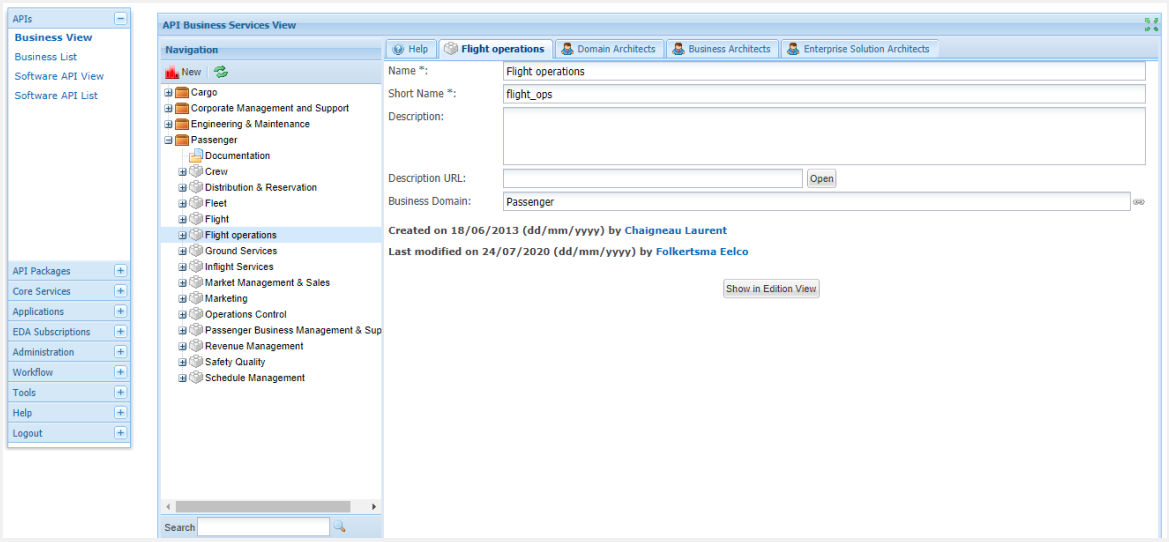
L'avènement des APIs a amené dans cette gouvernance un nouveau rôle, Responsable de la "productisation" et de l'évangélisation des APIs, donc pleinement impliqué dans l'élaboration de la stratégie avec le *Community Manager*.



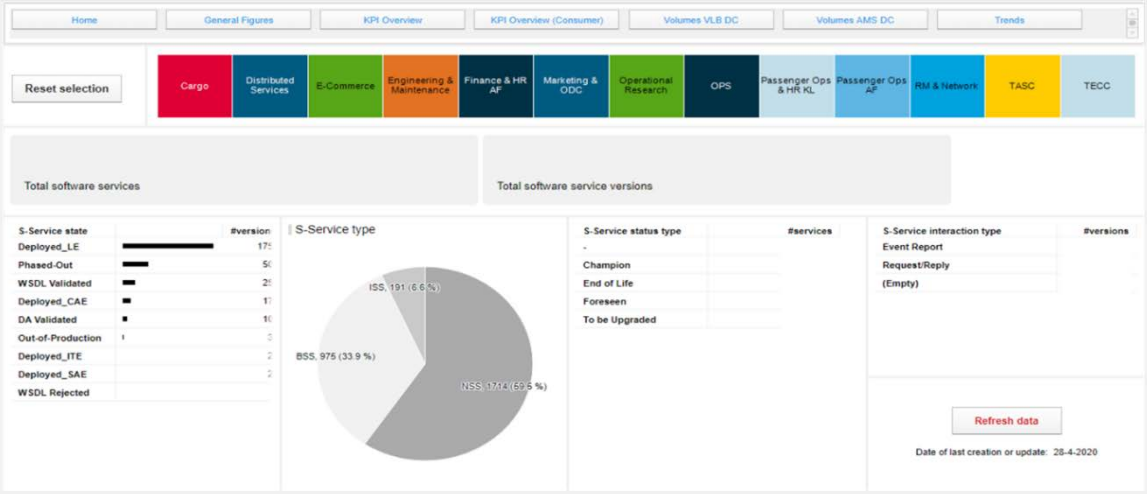
Ainsi le cycle de vie des services et APIs se déroule, jalonné de points de contrôles formels (API *Governance Board*, *Joint Architecture Committee*), lors desquels sont entérinés les décisions de création, évolution et utilisation de ces *assets*.



Afin de rendre la gouvernance la plus efficace possible, une approche self-service est en place, coordonnant les étapes de gouvernance au travers d'un outil central qui offre une visibilité complète sur le portefeuille de services/APIs et facilite le déroulement du cycle de vie de ceux-ci.



Cet outil permet en outre de produire des KPIs (Indicateurs de performances) relatifs à l'évolution, la consommation, la qualité du portefeuille de services/APIs et ainsi en assurer la cohérence.



Vincent Tassy, Air France-KLM

Le cas des microservices - L'avènement des architectures logicielles à base de microservices challenge les modèles historiques de gouvernance en introduisant des interactions entre services issus d'une même application, à très bas niveau, et dont la gestion au même niveau que les APIs « d'entreprise » ne fait pas sens.

La réflexion des entreprises se porte aujourd'hui sur la délégation de la gouvernance de ces microservices au niveau des équipes produits afin d'assurer la cohérence au niveau des domaines métier en appliquant le principe que tout franchissement de la frontière d'un domaine métier requalifie l'interface du microservice en API traditionnelle, qui retombe alors sous le coup de la gouvernance classique.

3.8 S'appuyer sur une solution d'API *management*

La servicisation du SI permet de partager, selon des conditions définies, les données de l'entreprise aux équipes internes, aux partenaires de l'entreprise et au public et de faciliter les interactions de façon contrôlée. La mise en place d'une solution d'API *management*, qui constitue un composant d'exposition des points d'entrée de l'API, est importante dans la réussite d'une architecture de servicisation.

En effet, l'étendue des fonctionnalités nécessaires pour délivrer des APIs de manière industrielle ne permet plus de construire des solutions maison. Il convient d'utiliser l'une des solutions API *management* du marché, en fonction des besoins et contraintes budgétaires.

L'attente première d'une solution d'API *Management* est l'exposition et la sécurisation des APIs au travers d'une *Gateway* (voire dans certains cas de plusieurs Micro-*Gateways*). L'accessibilité aux partenaires extérieurs demande de supporter une variété de standards de sécurisation établis (OAuth, OpenID/Connect, TLS, WS-Security, etc.). On sera donc vigilant sur les possibilités de la *Gateway*.

On peut, pour certains cas sensibles, désirer que la solution d'API *Management* prenne en charge la validation du contenu qui la traverse afin de centraliser cette opération et ne pas exposer inutilement ses *BackEnds* à du contenu malicieux. Cette fonctionnalité peut tout aussi bien être prise en charge par d'autres composants d'infrastructure de type pare-feu applicatif ou proxy.

Dans le cadre d'une exposition externe des APIs (mais pas uniquement), on voudra généralement être en mesure de mettre en place des quotas d'appels pour différents consommateurs. On peut aussi être intéressé par la possibilité de gérer des quotas globaux pour protéger ses *BackEnds*. Cela agit en complément du dispositif de sécurité dédié délivré par le fournisseur d'infrastructure pour se prémunir d'attaques, par exemple de typedéni de service.

Il est courant qu'une démarche de servicisation/APIsation mette en œuvre un portail Web dédié aux développeurs. C'est encore une des fonctionnalités offertes par les solutions d'API *Management*. On sera attentif à l'extensibilité du portail, son adhérence à l'ensemble de la plateforme d'API *Management* (Peut-on le remplacer par un développement maison s'il est jugé insuffisant ? Est-il basé sur un outil de gestion de contenu ou sur du pur contenu Web ?). Enfin les plateformes d'API *management* permettent nativement de gérer les besoins non fonctionnels (sécurité, haute disponibilité, scalabilité, résilience) par simple configuration au lieu de déléguer ces aspects à chaque développeur d'API (principe de « *configuration over development »).*

Nous avons vu que la mise en place d'une gouvernance autour des APIs demande d'être capable de produire des KPIs quant à l'utilisation des services. Il est donc primordial que la solution d'API *Management* fournisse les capacités de collecter les données d'usage, éventuellement de les présenter ou de les déverser dans son propre entrepôt de données afin de pouvoir bâtir des tableaux de bord pertinents.

Enfin, suivant les environnements on recherchera dans une solution d'API *Management* différents modèles d'hébergement, sur site ou dans le Cloud, géré en propre ou en mode SaaS. Certaines solutions du marché offrent des modèles hybrides sous diverses formes.

Les solutions d'API *Management* sont de plus en plus riches de fonctionnalités (éditeurs, outils de développement intégrés, référentiels de code, gestion du cycle de vie, etc.) qui peuvent s'avérer intéressantes en fonction de l'existant, tant en termes de technologie que de compétences.

4 Cadre *Open Digital Architecture* (ODA)

Lors de notre année de travail et pour mieux appréhender la mise en œuvre de la servicisation du SI, les participants ont souhaité avoir une présentation d'un cadre de travail. Ils ont choisi le cadre de travail ODA (*Open Digital Architecture*) utilisé dans le secteur des télécoms, même s'il s'ouvre aussi à d'autres secteurs aujourd'hui. L'objectif de ce chapitre présenté par Orange est d'introduire ce qu'est l'ODA, de préciser le contexte dans lequel les opérateurs réunis au sein du TM Forum l'ont élaboré et enfin de détailler les Open APIs de ce cadre.

4.1 Présentation générale

Le cadre ODA a séparé le système d'information global d'une entreprise en grands blocs listés ci-après et utilise des Open APIs pour les connecter entre eux.

Source : TM Forum

Figure 1 : Structure du *framework* ODA

La partie « *Engagement management* » gère l'ensemble des interactions avec les utilisateurs (personnes et systèmes) du SI en fonction de leur terminal et de leur canal d'échange avec l'entreprise. La partie « *System of Engagement* » ne couvre aucun processus métier ni données métiers qui sont dans la partie « *System of Record* ». Cette dernière contient les « *FrontEnds* » c'est-à-dire les logiciels qui gèrent l'interaction avec les utilisateurs du SI.

La partie « *System of Records* » porte les processus Métier appelant toutes les fonctions qui mettent à jour les données de référence ou activent les machines. Cette partie contient les *BackEnds* et les processus opérationnels.

La partie « *System of Records* » se divise en 3 domaines :

- « ***Party Management*** » s'occupe du « qui » et du « pourquoi ». Ce domaine du *System of Records* traite les personnes physiques ou morales impliquées dans des processus commerciaux. Il gère également la relation globale entre l'entreprise, ses clients et ses partenaires (CRM/PRM, gestion des interactions, identité, facturation, règlement des partenaires et des fournisseurs, avec la gestion des employés : RH, ...). Cette partie du *System of Records* n'est pas liée au catalogue commercial ou technique et peut s'appliquer à toute entreprise.
- « ***Core Commerce Management*** » s'occupe du « quoi ». Il traite de tous les processus et données liés au catalogue commercial (offres et produits). Il couvre les fonctions au niveau des offres et des produits : par exemple, la saisie des commandes, l'orchestration de la livraison des produits, la tarification (frais récurrents, tarification de l'utilisation selon les tarifs définis dans un plan de tarification...), la garantie des produits. Cette partie du *System of Records* est indépendante des clients et des moyens techniques.
- « **Production** » s'occupe du « comment ». Ce domaine outille l'ensemble des processus de fabrication, livraison, SAV, etc. Il porte tous les processus et données techniques, liés aux services ou ressources permettant de mettre des produits à disposition des utilisateurs. Cette partie du *System of Records* est également indépendante des clients et des offres commerciales

La partie *System of insight* contient les *BackEnds* analytiques travaillant sur toutes les données capturables et permettant d'aider à la décision (via interfaces *FrontEnd*) ou de démarrer automatiquement un processus opérationnel (du *system of records*). Seules sont créées par ces *BackEnds* des données calculées à partir des données de référence (KPI, comptes…). Les solutions d'intelligence Artificielle font partie de cette partie *system of insight*…. Mais aussi plus prosaïquement la comptabilité.

Les cadres gris *Decoupling & Integration* sont des APIs ou open APIs qui permettent d'articuler les blocs et offrent la possibilité d'ouvrir un certain nombre de données à des tiers pour leur usage business. Il y a des APIs en self-service un peu partout pour permettre les interactions entre les différents composants et les composants des différents blocs peuvent être utilisés en *plug and play* (prêts à l'emploi). Les APIs sont définies par le TM Forum. L'open API est utilisable par des composants logiciels réputés autonomes qu'ils soient ou non open source (cf microservices). Ce partage nécessite la documentation des APIs pour qu'elles soient effectivement utilisées. L'idée est de standardiser les Open APIs pour favoriser la relation soit interne à l'entreprise, soit externe avec ses partenaires et même d'autres écosystèmes.

La généricité est recherchée intrinsèquement. Beaucoup d'éléments de l'ODA sont génériques et utilisables dans d'autres business que ceux des entreprises télécoms, fournisseurs de services et intégrateurs (seul le bloc « production » est le plus souvent spécifique).

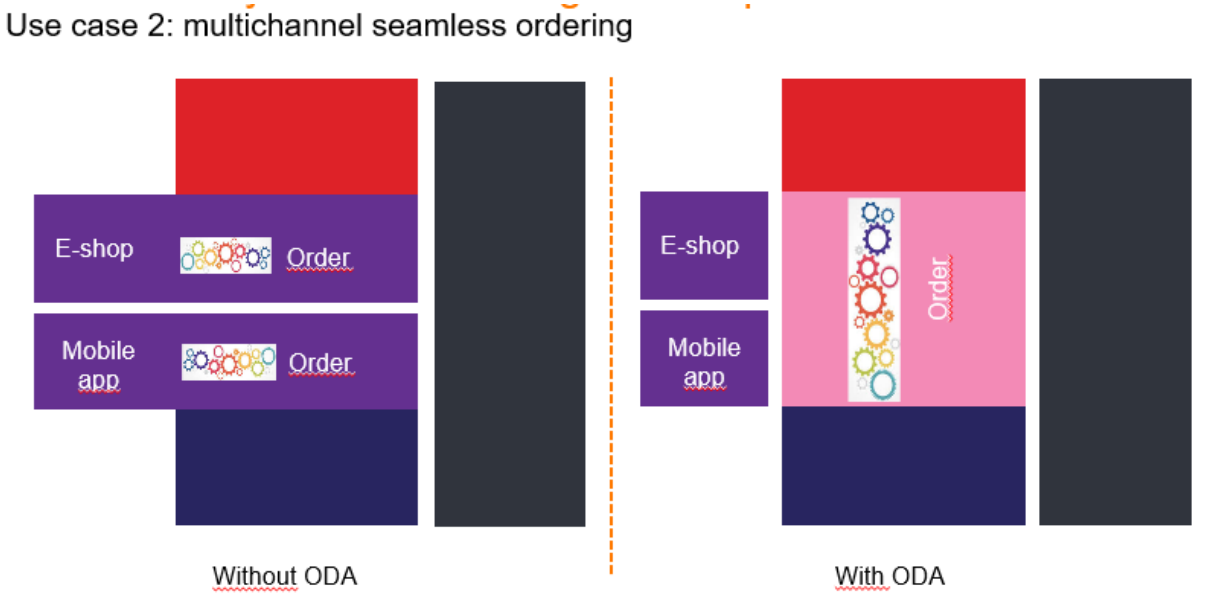
Ce découpage tel que présenté sur le schéma ci-dessus, représente déjà un défi pour la plupart des SI d'aujourd'hui. Il faut implémenter ce cadre progressivement. L'articulation de ces blocs assure l'intégration facile d'une solution d'un éditeur extérieur.

Le cadre ODA a **séparé les processus opérationnels des processus analytiques**. Pour la partie Client, l'ODA préconise de n'avoir qu'un seul répertoire de contacts pour l'ensemble des clients. Un client est une personne qui a plusieurs numéros de téléphone, d'adresses mail, etc… Souvent les services n'ont

qu'une partie des informations et donc considèrent qu'il s'agit de plusieurs clients ce qui ne facilite pas une expérience client unifiée. L'ODA a pour but de rationaliser les échanges mais il faut dans un premier temps identifier les chemins.

À la mi 2020, 54 APIs définies par le TM Forum ont déjà été déployées dans 66 pays. Cet engagement s'est traduit par un *ODA & Open APIs Manifesto* signé par une vingtaine d'entreprises (https://www.tmforum.org/oda/open-digital-architecture-open-api-manifesto/) qui œuvrent pour construire un cadre de travail ODA homogène et efficient. Le manifeste consiste en un ensemble de règles et de principes pour la conception de capacités technologiques modulaires et réutilisables ; une taxonomie et une définition communes des composants (les limites, les capacités et les capacités fonctionnelles et non fonctionnelles "minimales viables" et les APIs requises). Les signataires s'engagent à contribuer à une définition fine des APIs, à la construction de références pour faciliter leur utilisation et intégration.

Afin d'illustrer la valeur ajoutée qu'apporte l'utilisation du *framework* ODA dans son business, ses processus et son organisation, Orange a présenté au groupe de travail le cas d'usage pour supporter des offres multicanal. L'utilisateur commence son achat sur le site web, puis en sortant de chez lui continue avec l'application *smartphone* en reprenant là où il était. Comme il a une question, il appelle le service client qui finalise son achat. Cependant organiser une commande multicanal sans faille n'est pas si simple. Le schéma illustre pourquoi ce cas était difficile à traiter avant ODA, et comment il l'est facilement avec ODA.



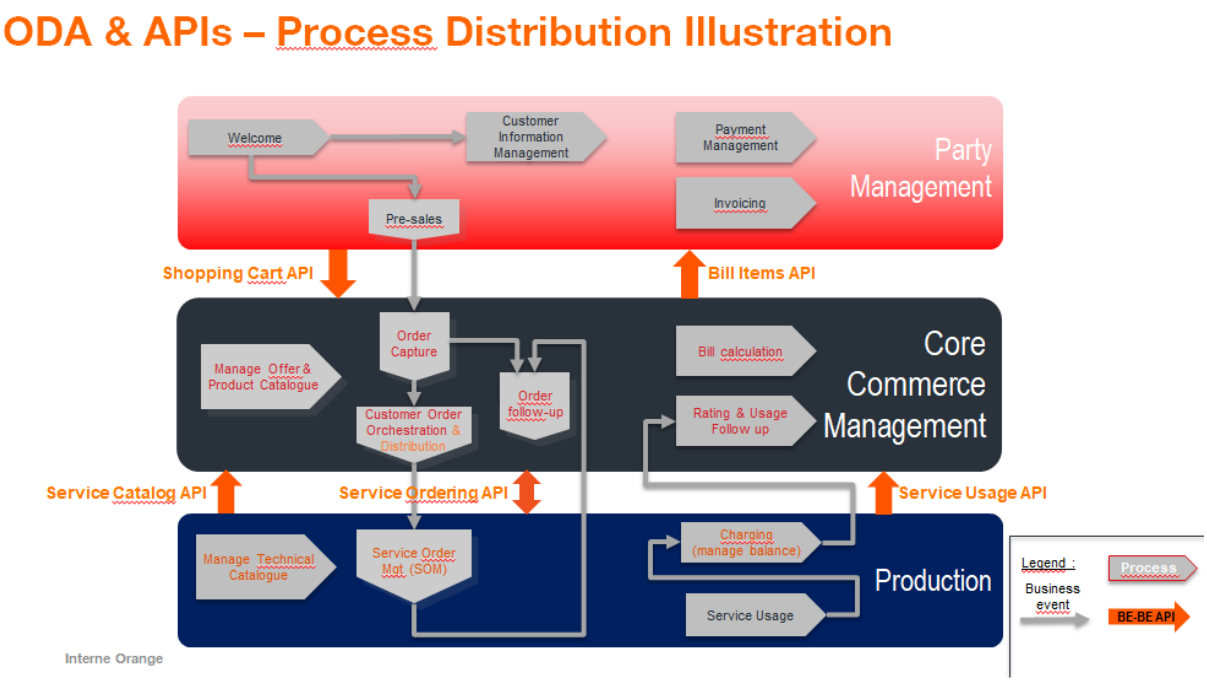
Source : TM Forum

Figure 2 : Apport de l'ODA dans une commande multicanal

Le cadre ODA sépare la partie interaction client, spécifique au canal ou terminal utilisé de la partie processus de commande générique. Il assure ainsi la multicanalité de ce processus.

4.2 Processus de distribution des APIs entre les grands blocs ODA

Processus de distribution des APIs :



Source : TM Forum

Figure 2 : Processus de distribution des APIs

Un premier type d'API concerne tout ce qui touche à la **publication d'information**. L'exemple donné est celui d'un catalogue de produits à commercialiser. Dans le domaine CCM (« *Core Customer Management* »), on traite les processus et les données du catalogue commercial, c'est à dire les offres et les produits. Dans le domaine production, on traite du catalogue technique qui décrit les savoir-faire (ou services) de l'entreprise et comment ils sont réalisés. Les savoir-faire (ou services) disponibles sont publiés par le domaine Production vers le domaine CCM afin de décrire les produits à partir de ces savoir-faire (*Service Catalog* API).

Les processus liés à la relation client (gestion d'information client : CRM, identité du client, facturation) sont gérés dans le domaine du *Party Management*. Lorsqu'un acte d'achat est engagé avec une potentialité de vente, on enclenche un processus de prise de commande. L'événement métier est transmis à *Core Commerce Management* pour déclencher la prise de commande (*Shopping Cart* API). L'achat effectif déclenche la facturation ou le paiement immédiat du client puis avec un autre type d'API, l'API *service ordering*, la livraison du produit et son suivi. La valorisation de l'offre dépend du produit livré ou utilisé et des conditions de commercialisation de ce produit : par exemple le type de forfait, temps d'appel écoulé, etc. L'information de commande puis d'usage permet de déclencher à chaque étape la valorisation qui peut être aussi périodique (abonnement). Une nouvelle API, le *bill items* API est nécessaire pour produire tous les éléments facturés : la facture indique pour chaque ligne valorisée le contrat (ou « offre installée ») et les usages des produits liés au contrat donnant lieu à un montant dû sur une période donnée. Ensuite la facture doit être produite selon les processus et règles

de présentation de facture. La facture est disponible via l'API *Customer Bill*. Elle sera ensuite présentée au payeur selon des règles de type « *Party Management* ».

Les choix de distribution dans les différents blocs définissent le cadre d'utilisation des APIs.

Les données sont distribuées dans les différents blocs du « *party management* » qui gère les processus des organisations et des personnes (interactions, comptes, facturation, confidentialité, etc.) et du « *core commerce management* » (données du catalogue, de la commande et du parc pour les produits et les offres).

Processus et articulation de l'interface graphique :

Lorsqu'une activité ne peut pas être automatisée, une interaction avec le *FrontEnd* est alors nécessaire. Il faut donc que soit précisée l'information attendue au niveau du *FrontEnd*. Le type d'API, *Process Flow API* (en français API de déroulement du processus) qui entre en jeu, est différent de ceux vus précédemment car cette API a pour vocation d'orchestrer les échanges entre couches process et cinématique IHM. Les *process flow APIs* servent de guide au système d'engagement pour demander à l'utilisateur ce qui est attendu par le processus. C'est son contexte d'utilisation à des fins d'orchestration qui est singulier.

Les API utilisées dans le cadre ODA sont soit des *core* API, soit des API *process* (avec la gestion des événements associés) qui orchestrent globalement les processus.

Conclusion

Afin de répondre aux attentes nouvelles de leurs clients, aux enjeux business poussant à l'exploration de nouvelles opportunités, les grandes entreprises et administrations publiques ouvrent leur architecture d'entreprise, leur système d'information, aux équipes internes ou à leur écosystème, proche ou étendu. Elles le font au travers de services techniques ou fonctionnels, idéalement conçus à partir d'un cadre d'architecture global et rendus accessibles par le biais d'APIs. Afin d'assurer la cohérence de ce portefeuille de services, les entreprises ont recours à des règles bien définies et appliquent plusieurs principes clés : **modularité**, **ouverture**, **sécurité** et **segmentation.**

Cela nécessite d'acculturer l'ensemble des collaborateurs aux bonnes pratiques : préciser les termes et les aspects techniques, standardiser les développements, les partager en interne, avoir le réflexe de regarder ce qui existe déjà**,** maximiser la réutilisation des APIs, etc.

La prise en compte de ces bonnes pratiques et faire preuve de pragmatisme permettent d'accélérer les démarches d'agilité des entreprises et d'envisager à terme la mutualisation de certains processus métiers ou de développements communs à leur secteur d'activité ou à leur écosystème. Ainsi, en mutualisant ces composants non différenciants, les entreprises et administrations publiques peuvent se concentrent sur leur cœur de métier et là où elles apportent de la valeur.

Avec l'émergence de l'IoT, certains produits industriels deviennent source de services. Cela nécessite d'étudier toute la chaîne de valeur possible, apportée par les composants, équipements et/ou appareils connectés afin d'identifier de nouvelles opportunités business avec ces services. Il est donc important, selon Wavestone, de chercher à rendre les composants les plus indépendants avec des interfaces de connexions uniformisées et architecturées permettant d'offrir la modularité et de faire évoluer les services rapidement.

Alors que la servicisation tend à répondre à des besoins de tout premier ordre pour les entreprises et organisations publiques, facilitant l'intégration à leur écosystème, peu ont pour le moment poussé cette démarche jusqu'à la commercialisation de ses actifs au travers d'une monétisation des APIs et d'une facturation à l'usage à l'instar de ce qui est proposé par les fournisseurs de services Cloud/SaaS. Cette facturation à l'usage est pourtant souvent supportée par les plateformes d'API *Management* du marché. L'exploitation de cette ressource de manière commerciale, au travers de plateformes métier consolidées ou de places de marché pourraient bien être la prochaine phase d'évolution de ce paysage en plein développement.

Annexe

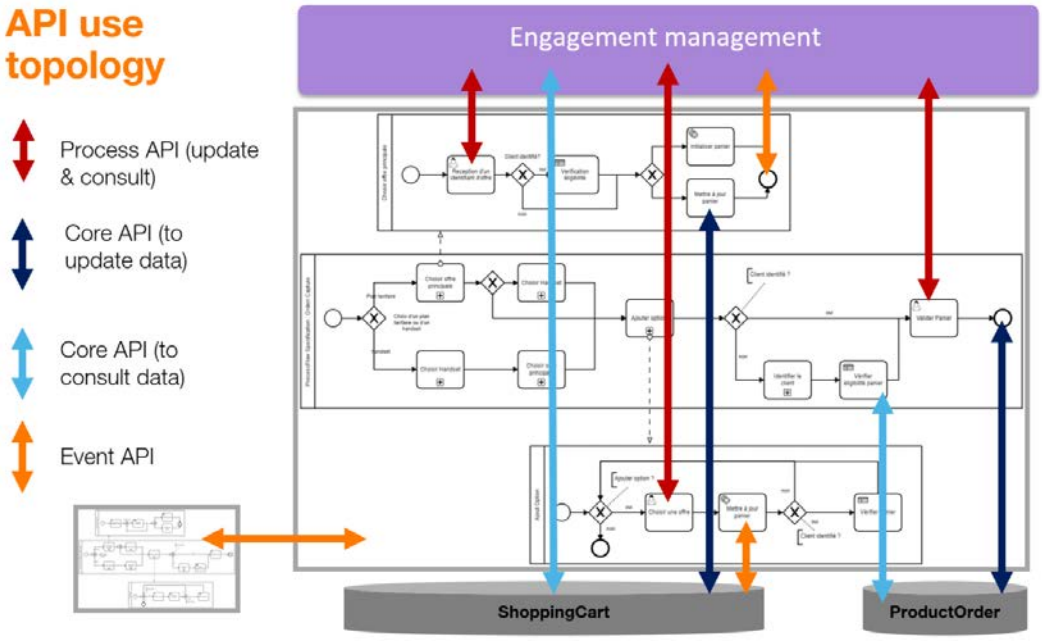
L'annexe développe les deux grands types d'API utilisées dans le cadre ODA : les *core* APIs et les APIs *process* (avec la gestion des événements associés) qui orchestrent globalement les processus.

Il existe tout d'abord les APIs de gestion des données (les *Core* APIs). Ces APIs permettent de consulter et de mettre à jour les données. Les opérations de consultation sont accessibles à tous les composants du SI sous réserve de gestion des droits. Les opérations de création et modification sont réservées aux composants de *BackEnd* en charge des exécutions des process métiers.

Il y a ensuite l'API propre à la gestion du processus métier : il s'agit de l''API *process flow* qui fonctionne en question/réponse avec le *FrontEnd* (données saisies permettant de faire avancer le processus et de mettre à jour les données). Elle est utilisée pour les interactions entre *FrontEnd* et *BackEnd* pour fournir les informations attendues de l'utilisateur par le processus (traitement des tâches manuelles). Elle est également utilisée entre *BackEnds* pour déclencher un processus d'un autre bloc ODA, par exemple gérer la création du client (bloc *Party*) lors d'une prise de commande (bloc Commerce)

Enfin il faut citer l'API de publication d'évènements qui va permettre à toutes ces APIs de déclencher des événements.

L'ensemble de ces APIs est fourni par les 4 blocs ODA couvrant les processus opérationnels et analytiques.



Source : TM Forum

Figure 4 : Topologie des APIs

Core APIs

Les *core APIs* permettent de rechercher, consulter et gérer les données. Elles isolent l'utilisation des données de la complexité du système sous-jacent. Il n'y a qu'un seul moyen d'accéder à la donnée. Les *core* APIs sont construites une fois pour plusieurs utilisations dans des projets divers et distincts. Ainsi, la modernisation d'une API ne doit pas poser de problème aux systèmes consommateurs car ils ne se rendent pas compte du changement.

C'est dans la réutilisation d'API que l'on trouve des gains de simplification. Il est important de veiller à avoir une cohérence globale des *core API* les unes par rapport aux autres. C'est par exemple le cas des APIs catalogue, commande et parc qui doivent être cohérentes. Les données gérées peuvent être statiques, comme des données d'inventaire produit, de catalogue, ou dynamiques, basées sur des transactions comme la prise de commande ou un ticket d'incident.

Une liste d'APIs déjà conséquente a été définie au TM Forum avec la ligne de conduite de travailler de façon agnostique par rapport au secteur d'activité.

Le *FrontEnd* a toujours la possibilité de consulter les données clients, de chercher le catalogue, avec les fonctions *get* des APIs *Core*.

Derrière chaque API, est défini un modèle de ressource (données manipulées). Sur chaque ressource gérée par API (commande, client, facture) on définit des événements : création de la commande, changement de la commande, etc. Les notifications envoyées aux applications abonnées sont construites à partir des événements. Les APIs sont typiquement définies par des spécifications OpenAP, lisibles aussi bien par les humains que par les machines et faciles à utiliser.

Process APIs

Le rôle des « *process* API » est de suivre les étapes d'un processus. Chaque processus est défini une seule fois dans un seul *BackEnd*. L'API *process* est utilisée pour gérer une instance d'exécution d'un processus métier, via un ensemble de tâches automatiques et manuelles. Les « *process* APIs » ont été construites de façon à pouvoir interagir avec un ou de multiples systèmes pour l'exécution d'un processus. Un mécanisme permet de décrire les données attendues pour une tâche requérant une saisie par un utilisateur (description d'un formulaire). Tout au long du processus, c'est le processus qui utilise les *core APIs* pour la mise à jour des données. Les *process APIs* gèrent la complexité des données en silos (multiplicité et dépendance). L'objectif est de cacher du *FrontEnd* la complexité d'un processus requérant la mise à jour en cohérence d'un ensemble de ressources différentes. Le processus de bienvenue est utilisé pour proposer les actions possibles en fonction du canal utilisé par le client : le *chat*, le web, etc. Ce processus déclenchera ensuite le processus adéquat en fonction du choix effectué. Les *process API* séparent la façon dont les données sont stockées de la façon dont elles sont capturées. L'API est le vecteur qui permet d'échanger entre le *FrontEnd* et les process exécutés depuis le *BackEnd*.

La logique business spécifiant les processus et les tâches est définie une fois dans le *BackEnd* et non spécifiquement dans chaque *FrontEnd*. L'instance de processus peut être créée suite à une interaction *FrontEnd* ou à l'exécution d'un autre processus/tâche - directement ou indirectement.

L'API *event* gère la publication des événements. Par exemple une fois le process exécuté, il envoie un événement pour préciser qu'il est fini. Un composant spécifique de gestion des évènements peut être utilisé pour chorégraphier des appels d'APIs dans les *BackEnd*/ *FrontEnds* en fonction des évènements reçus et par exemple déclencher de nouveau processus.

Le *BackEnd* doit fournir toutes les informations pertinentes au *FrontEnd* pour lui permettre de compléter l'instance de la tâche (par exemple, fournir le formulaire de données, fournir la valeur de la liste de sélection, etc.). Le processus business est constitué d'une suite de tâches définies avec des règles de contrôle et d'enchainement. Les tâches sont soit complétées automatiquement, soit manuellement.

Le *FrontEnd* et le *BackEnd* doivent pouvoir disposer d'un modèle de communication permettant des échanges synchrones mais aussi un modèle basé sur les événements afin de ne pas contraindre les échanges synchrones. L'API *process* génère aussi des événements (via l'API *event*) pour permettre l'exécution des tâches d'un processus de manière asynchrone.

Les processus sont découpés en tâches. L'hypermedia, hyper lien vers des ressources ou sur elle-même, prend son sens lors d'un processus instancié, en précisant quelles informations renvoyer, comment et où. Un processus est géré via une API générique ou spécialisée : l'API *ProcessFlow* expose des *resources processFlow* & *taskFlow* qui peuvent être spécialisées respectivement en *OrderCapture* et *SelectMainOffer* par exemple).

L'enchaînement des processus doit être dynamique y compris entre les différents blocs du *System of Records* :

- Lorsqu'un nouvel utilisateur se connecte et effectue une première commande, alors le processus de création de client doit être enclenché avant la validation de la commande (ce processus de création de client n'étant pas exécuté pour un client déjà connu).
- Le processus « prise de commande » une fois terminé publie un évènement qui déclenche le processus de livraison.

Quand une tâche est finie (de façon nominale ou non), elle émet un *event*. Chaque API – *core* ou *process* - définit et déclenche des événements.

Au service de la croissance économique et de la compétitivité de nos membres, grandes entreprises et administrations publiques françaises, utilisatrices de solutions et services numériques, par la réussite du numérique

Le Cigref est un réseau de grandes entreprises et administrations publiques françaises qui a pour mission de développer la capacité de ses membres à intégrer et maîtriser le numérique. Par la qualité de sa réflexion et la représentativité de ses membres, il est un acteur fédérateur de la société numérique. Association loi 1901 créée en 1970, le Cigref n'exerce aucune activité lucrative.

Pour réussir sa mission, le Cigref s'appuie sur trois métiers, qui font sa singularité.

1/ Appartenance :

Le Cigref incarne une parole collective des grandes entreprises et administrations françaises autour du numérique. Ses membres partagent leurs expériences de l'utilisation des technologies au sein de groupes de travail afin de faire émerger les meilleures pratiques.

2/ Intelligence :

Le Cigref participe aux réflexions collectives sur les enjeux économiques et sociétaux des technologies de l'information. Fondé il y a près de 50 ans, étant l'une des plus anciennes associations numériques en France, il tire sa légitimité à la fois de son histoire et de sa maîtrise des sujets techniques, socle de compétences de savoir-faire, fondements du numérique.

3/ Influence :

Le Cigref fait connaître et respecter les intérêts légitimes de ses entreprises membres. Instance indépendante d'échange et de production entre praticiens et acteurs, Il est une référence reconnue par tout son écosystème.

www.cigref.fr

21 av. de Messine, 75008 Paris

+33 1 56 59 70 00

cigref@cigref.fr