SOA
MICROSERVICES, API MANAGEMENT

SOA

MICROSERVICES, API MANAGEMENT

Le guide de l'architecture d'un SI agile

Xavier Fournier-Morel

Pascal Grojean

Guillaume Plouin

Cyril Rogon

5e édition

DUNOD

Image de couverture : © ThomasSereda, iStock

© Dunod, 2020
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-080738-3

Table des matières

Partie 4
SOA et architecture d'entreprise

Avant-propos

Il y a 10 ans, on présentait les architectures de services, dites SOA, comme une solution formidable pour mettre fin aux problématiques d'intégration entre applications hétérogènes. SOA devait aussi rendre les systèmes d'information plus adaptables et plus agiles.

SOA a initialement déçu certaines entreprises car les grands éditeurs ont proposé des solutions « tout en un », capables de gérer l'ensemble des problématiques d'intégration, au travers d'une plate-forme technique, généralement onéreuse car complexe, et vendue via un discours du type : « achetez d'abord l'outil et SOA viendra après sans effort ».

De leur côté, les géants du Web (Google, Amazon, Facebook…) n'ont gardé de SOA que quelques patterns d'architecture qu'ils ont appliqués de manière pragmatique, au travers d'architectures simples, basées sur les technologies du Web. Ils ont ainsi su bâtir des systèmes d'information parmi les plus exigeants en termes de performance en se basant sur des architectures de services. Ils ont réussi tout au long des dix dernières années là où d'autres ont échoué.

L'expérience des géants du Web démontre la pertinence des architectures de services, mais elle démontre aussi l'agilité que peuvent offrir ces architectures, en permettant de transformer un SI à un rythme très rapide. C'est d'ailleurs cette rapidité qui est au cœur de la disruption de nombreux secteurs d'activité (taxis avec Uber, hôtellerie avec Airbnb, etc.).

Il faut néanmoins noter que si les géants du Web gèrent des volumes colossaux d'utilisateurs, de données et de requêtes, leurs règles métiers sont plus simples que celles des contrats d'assurance ou celles des systèmes de *yield management* des compagnies aériennes. Il est donc nécessaire de prendre du recul et d'adapter leurs patterns d'architecture au contexte des entreprises dotée d'un SI existant et complexe.

Le propos de cet ouvrage est donc de présenter d'une part les concepts, les méthodes, les technologies et les patterns qui permettent aujourd'hui de planifier, construire et maintenir une architecture de services opérationnelle en entreprise. On abordera en particulier les microservices, les architectures réactives, les gestionnaires d'API. Les auteurs, qui ont plus de 25 ans d'expérience dans les architectures informatiques, gardent ici un œil critique sur les nouvelles technologies trop souvent présentées comme révolutionnaires, et choisissent de donner un éclairage sur ce qui est à prendre et à laisser dans ces architectures pour un contexte d'entreprise.

- ***Première partie – Le cahier des charges des SI agiles***

L'objectif de cette première partie est d'introduire les problématiques de rationalisation des systèmes d'information et la tension créée par la transformation digitale.

Elle présente tout d'abord un historique de l'évolution des systèmes d'information et montre comment ces derniers sont souvent complexes et peu cohérents. Elle évoque ensuite le besoin d'« agilité » des SI dans le contexte actuel de disruption générale liée à la digitalisation du monde[1].

Enfin, elle aborde les exigences métiers et techniques que l'on peut attendre d'une nouvelle démarche d'organisation et d'intégration du SI, que ce soit dans le cas d'une volonté d'innovation ou de meilleure maîtrise de celui-ci. La suite de l'ouvrage montrera comment SOA répond à ces exigences.

Cette partie introductive est accessible à tous les profils : maîtrises d'œuvre comme maîtrises d'ouvrage.

◆ *Seconde partie – Les concepts & l'approche SOA*

L'objectif de cette deuxième partie est de clarifier les concepts utilisés, en commençant par la notion de services et d'architecture de services, puis en introduisant le concept de microservices. Elle présente une architecture de référence positionnant les différents concepts ainsi que l'évolution de leur maturité à travers trois générations du style SOA. Elle clarifie également pourquoi les Web Services sont utiles, mais pas indispensables.

Elle s'attache à décrire les fondements du concept de service et notamment la notion de contrat et de messages.

Enfin, elle introduit les socles qui peuvent être mis en place de façon progressive en regard de l'architecture SOA de référence, notamment les socles de services et d'intégration. On y aborde en particulier l'émergence des gestionnaires d'API, concept clé des expositions de service et d'API et outil qui facilite l'innovation au sein d'un SI. L'utilité et la complémentarité des socles tels que le bus de services, le gestionnaire de flots d'événements, le gestionnaire de processus ou le référentiel de gouvernance des services sont également démystifiés.

Cette partie est conceptuelle. Elle est accessible à tous les profils de maîtrises d'œuvre, mais aussi aux maîtrises d'ouvrage qui désirent appréhender en profondeur les architectures fonctionnelles réalisables avec SOA.

◆ *Troisième partie – Méthodologie SOA*

La partie 3 présente les principaux aspects méthodologiques de l'approche SOA avec deux points de vue complémentaires : le métier et la technique.

---

1. "*Software is eating the World*", selon Marc Andreessen, ancien patron de Netscape.

Cette partie traite d'abord de l'identification d'un périmètre pour SOA et des services candidats en utilisant l'exemple d'une entreprise fictive, Fil Rouge. Vient ensuite la modélisation des services et de l'architecture. Elle présente la manière de traiter aussi bien la définition classique d'un service par requête-réponse, d'un microservice asynchrone que celle de services liés aux objets connectés via des flots d'événements à fortes contraintes de « réactivité ». Cette modélisation s'appuie sur la présentation de patterns architecturaux que l'on considère comme éprouvés sur le terrain. Elle aborde ensuite la modélisation des processus métier, l'adoption de SOA par les entreprises étant en grande partie liée à la volonté de mettre en place de nouveaux processus métier plus électroniques. Enfin, le chapitre sur la modélisation des applications interactives propose d'étudier l'impact de SOA sur le classique modèle MVC et ses variantes.

Cette partie méthodologique vise les architectes et responsables de solutions.

- ***Quatrième partie – SOA & architecture d'entreprise***

Le recours aux architectures de services apporte des réponses bien au-delà d'un seul projet ou même d'une seule solution multi-applicatives de l'entreprise. C'est là une de ses plus grandes forces. Les initiatives raisonnant à l'échelle des points de vue de toutes les parties prenantes du SI (stratégie, métier, technique, opérationnel, voire clients) apparaissent aujourd'hui comme une clé de construction cohérente d'une architecture à l'échelle de l'entreprise. Plusieurs cadres de description comme Zachman, TOGAF, DODAF, NAF, etc. se proposent pour standardiser un panel plus ou moins large de ces points de vue. Pourtant, il est intéressant de comprendre pourquoi tous portent progressivement leur attention sur SOA pour servir de liant ou de catalyseur à la mise en cohérence des éléments à construire et aligner au sein de l'entreprise (performance, processus métiers, modèle des données de référence, solutions applicatives, paquetage de logiciels, ressources d'infrastructure, etc.).

Les outils de construction et de gouvernance à l'échelle de l'entreprise apparaissent pour supporter cette structuration, via l'analyse et le suivi du fonctionnement et des transformations propres à chaque entreprise. L'approche SOA permet alors de fournir la colonne vertébrale d'un référentiel unifié de cette organisation.

Cette partie organisationnelle et le cadre de démarche s'adressent aux architectes d'entreprise, responsables de lignes métiers ou de directions informatiques.

- ***Cinquième partie – Robustesse et performance SOA***

L'émergence des microservices met en évidence le caractère de plus en plus distribué du système d'information. Sa robustesse et sa performance deviennent des points

clés non seulement pour les directions de la production des DSI, mais aussi pour les architectes et les développeurs impliqués dans une démarche DevOps. Il n'est plus question pour aucune direction de rejeter ces problématiques « vers l'autre » ou « à plus tard » dans un monde dont le système nerveux devient digital.

Ces exigences de robustesse et de performance ne sont pas triviales à satisfaire, en particulier pour les services dits « à état » (*stateful*). L'exigence de robustesse d'un service est présentée dans cette partie comme la capacité technique de ce service à être disponible sans interruption. L'exigence de performance est quant à elle abordée comme la capacité technique de ce service à absorber une charge de travail pouvant varier fortement. Cette partie présente alors les différentes solutions possibles pour satisfaire ces exigences. Elle propose également une réflexion sur les problématiques techniques au cœur de ces solutions, en particulier le fameux théorème CAP et ses conséquences sur la réplication des données, la tolérance aux pannes réseau et la cohérence des transactions. La compréhension de ces problématiques est indispensable pour effectuer son choix parmi les outils de plus en plus nombreux dans ce domaine, puis pour déployer et configurer ces outils.

Cette partie s'adresse aux architectes techniques aussi bien qu'aux développeurs et aux responsables de la production soucieux de travailler ensemble pour garantir le bon fonctionnement des services soumis à de plus en plus fortes contraintes.

- ***Sixième partie – Monter en puissance sur SOA, microservices et gestion d'API***

On aborde ici les aspects techniques de la mise en pratique des architectures de services : les normes, langages, technologies, outils et bonnes pratiques qui sous-tendent tout le cycle de réalisation de services et microservices, depuis la conception, le développement, le déploiement jusqu'à la sécurisation et la gestion de l'exploitation d'interfaces publiques, en passant par le recours à des socles d'infrastructure pour industrialiser les mises en œuvre SOA.

Cette partie s'adresse aux équipes de réalisation, concepteurs, développeurs, architectes techniques et responsables d'exploitation de services.

- ***Septième partie – Choisir son parcours SOA***

Une démarche vers SOA ne part jamais de zéro, d'autant que nombre d'entreprises ont débuté une démarche SOA 1.0 il y a quelques années. En fonction de la situation du SI de l'entreprise, cette partie propose un cheminement pragmatique, selon des étapes et jalons clés, évitant le syndrome du mur infranchissable et du Big bang explosif.

Cette partie est destinée avant tout aux directions informatiques et aux responsables de projet.

◆ *Compléments sur le Web*

Les auteurs souhaitaient ajouter quelques extensions à cet ouvrage, sans le rendre trop volumineux : des exemples de codes pratiques et informations complémentaires. Pour faciliter leur mise à jour régulière, ces éléments sont rendus disponibles sur le Web à l'adresse https://www.dunod.com/sciences-techniques/soa-microservices-et-api-management.

◆ *Remerciements*

Les auteurs tiennent après leur quatrième édition de cet ouvrage à remercier une nouvelle fois, et encore plus chaleureusement leurs compagnes pour leur immense patience et soutien pendant les périodes de rédaction et remise à jour de ce livre.

Leur reconnaissance va aussi à leurs entreprises respectives qui ont choisi de sponsoriser et rendre possible cette nouvelle édition de l'ouvrage.

Enfin, ils remercient très vivement leurs collègues et amis qui ont bien voulu relire le manuscrit initial et les faire bénéficier de remarques ou contacts pertinents, en particulier, Monsieur Claudio Parnenzini, urbaniste visionnaire en charge des Systèmes d'information de l'État de Fribourg (Suisse), Monsieur Dominique Debailleux, architecte senior SOA enthousiaste chez Emoxa et Monsieur Damien Engrand, architecte technique SOA chevronné et consultant chez Emoxa (France).

Préface

L'architecture orientée-service, plus connue sous le nom de SOA selon l'acronyme anglais, est une expression incontournable dans le monde des systèmes d'information depuis le début des années 2000. Il existe quelques bons livres, dont celui-ci, qui expliquent ce qu'est SOA : SOA est une approche fondamentale pour maitriser le système d'information. En revanche, je ne connais pas d'autres livres qui expliquent le « comment déployer SOA » avec autant de pertinence, de précision et de talent. C'est pourquoi j'avais fait de ce livre une des pierres angulaires du cours « Théorie et pratique du système d'information » que je donnais à l'Ecole Polytechnique il y a quelques années.

Voici une nouvelle édition en 2017 qui a suivi les changements majeurs des dernières années, liés à l'accélération du temps métier. Les systèmes d'information d'aujourd'hui, en particulier dans leur partie « digitale », sont en mouvement perpétuel et le code n'est plus un composant stable de leur fabrication, mais un flux permanent. Les frontières doivent devenir poreuses, l'ouverture des services est un prérequis. Très logiquement, cette nouvelle édition est considérablement enrichie dans la dimension API (interfaces), micro-services et distribution sur le *cloud*. Une des marques de fabrique de ce livre est l'utilisation des *patterns* (« motifs de conception ») qui font le trait d'union entre la compréhension des problèmes – comme par exemple les problèmes de performance des architectures cloud – et la mise en œuvre des solutions – telles que le pattern CQRS du chapitre 8.

Commençons par rappeler les enjeux qui justifient l'architecture orientée service. Le premier enjeu est « simplement » la réduction des coûts, en s'appuyant sur la réutilisation et la mutualisation. C'est l'enjeu le plus concret et le plus visible : il s'agit de construire un patrimoine de services partagés dans l'entreprise, réutilisés le plus souvent qu'il est possible. Le deuxième enjeu est lié, c'est celui de l'alignement stratégique du système d'information, qui est la clé de l'agilité au travers de l'anticipation. Le catalogue de services à produire doit être « prêt pour le futur », il représente le potentiel de situation du système d'information. Seule la co-construction avec les directions métiers et utilisatrices permet de construire un catalogue « le mieux aligné possible » avec les enjeux présents et à venir de l'entreprise. Le troisième enjeu de SOA touche à la maîtrise de la complexité du système d'information. Autrement dit, comment répondre aux besoins d'aujourd'hui sans construire une « usine à gaz » tellement complexe qu'elle réduit considérablement les chances de pouvoir répondre demain aux nouveaux besoins.

Il ne suffit pas de construire un catalogue de services pour faire du SOA, il faut construire des « services orientés-architecture ». Autrement dit, il faut construire les bonnes pièces du lego, ce qui est difficile, et c'est précisément pourquoi le livre que vous tenez entre les mains est important. Pour que SOA fonctionne, il faut construire

des services abstraits, modulaires et recomposables. L'analyse fonctionnelle traditionnelle que l'on pratique depuis 50 ans doit être intégrée au sein d'une modélisation des objets métiers et des processus métiers. La modularité, une propriété fondamentale du système d'information puisqu'elle gouverne l'agilité et la maîtrise des coûts (via la réduction des impacts), s'obtient au moyen d'une grammaire de services « bien découpés » (il s'agit plus d'un art fondé sur l'expérience que d'une science). La mutualisation et la réutilisation sont fortement liées à l'abstraction. Des services trop abstraits coûtent chers en adaptation et spécialisation, tandis qu'une abstraction trop faible réduit les possibilités de mutualiser. Pour reprendre une citation de ce livre, l'adoption de DevOps ne signifie pas qu'on puisse se passer d'architecture.

Une fois le modèle des services esquissé, il reste encore à les implémenter, ce qui réserve quelques surprises. Pour en citer quelques-unes, commençons par l'architecture de données. SOA ou non, les grands systèmes d'information reposent sur des architectures distribuées, qui posent des problèmes classiques de synchronisation et de cohérence. Ce livre aborde les questions importantes de distribution des objets métiers. SOA introduit une granularité et une distribution des services, ce qui pose des problèmes de performance (SOA n'est pas « *scale-free* », on ne peut pas décomposer et distribuer n'importe comment les traitements sans souffrir de problèmes de performance). Les contrats de services et l'exploitation des services sont des sujets essentiels que ce livre aborde. Il donne également des conseils précieux pour bien utiliser les différents outils et les différents paradigmes techniques qui servent à l'implémentation de SOA.

La dimension dynamique – voir le SI comme un processus continu de fabrication de services et non pas comme un catalogue – est essentielle aujourd'hui. Seule une organisation de « flux » permet d'atteindre l'agilité et de développer la capacité à profiter de la vague permanente de l'arrivée des nouvelles technologies et des opportunités associées. Si le logiciel est un flux, le code redevient important, ainsi que la culture de « *software craftsmanship* ». Ce livre montre le code, avec modération, parce que SOA est en premier lieu une pratique. L'accélération de cette dynamique a conduit à proposer l'approche microservices, qui consiste à unifier le service exposé avec le système technique qui le délivre, pour obtenir une complète autonomie qui favorise modularité, agilité et scalabilité. On trouve également un chapitre sur la « SOA réactive », qui détaille les fondamentaux et la mise en œuvre d'une architecture dynamique organisée autour des événements (EDA : *Event-Driven Architecture*). Cette mise en correspondance permanente, dans cet ouvrage, des principes, de la pratique et de la mesure est essentielle car une SOA dynamique est construite à partir de l'usage, de façon adaptative.

Une stratégie informatique moderne se décline forcément autour de ses interfaces, internes et externes, avec le terme devenu incontournable d'API (*Application Programming Interface*). L'utilisation interne est tournée vers l'agilité et la réutilisation, tandis que l'utilisation externe sert à exposer ses services au travers de partenaires pour étendre sa distribution et à enrichir ses propres produits avec les services et compétences d'autres partenaires. Cette combinaison/ recombinaison doit se

produire partout, tout comme la mesure et l'analyse sont, au 21e siècle, des bonnes pratiques que l'on retrouve dans tous les composants du système d'information. Ce livre est un guide pratique qui aborde tous les sujets clés comme la sécurisation, la gestion des versions ou la performance. Il traite à la fois la technique et la culture, puisqu'il rappelle au chapitre 22 qu'il faut « traiter les développeurs comme ses clients ».

Ce livre me semble essentiel pour réussir sa transformation digitale. L'expérience enseigne qu'il y a trois conditions à respecter :

- ✓ Adapter sa stratégie à ses moyens, son potentiel de situation, au lieu de continuer à penser d'abord et exécuter ensuite.
- ✓ Savoir « cultiver » ce potentiel (depuis les compétences des équipes jusqu'à l'agilité du SI) de façon émergente, sans « tirer sur la tige de la plante pour la faire grandir plus vite ».
- ✓ Pouvoir profiter de l'innovation externe et construire un SI ouvert qui permet à l'organisation de profiter de la richesse exponentielle des écosystèmes logiciels.

Ce livre donne les clés de chacune de ces étapes. Premièrement l'approche SOA est précisément une approche de co-construction de la stratégie et des moyens. Deuxièmement, ce livre insiste sur la mise en œuvre, les difficultés et la pratique car « c'est en forgeant qu'on devient forgeron » et c'est en utilisant les outils modernes et les piles logicielles ouvertes du cloud que l'on développe son potentiel d'agilité. Les compétences DevOps s'apprennent en pratiquant avec les bons outils. Troisièmement, ce livre donne les clés pour réussir conjointement à déployer une stratégie d'ouverture par les API et à ouvrir les capacités internes pour les interfacer avec les services externes.

Je ne connais pas de recette simple pour réussir la transformation nécessaire des systèmes d'information pour permettre aux entreprises d'affronter un monde incertain, mais je suis persuadé que « SOA, microservices, API Management» est un des meilleurs outils disponibles pour réussir.

Yves Caseau
Group Chief Information Officer chez Michelin
Membre de l'Académie des technologies, Président de la commission TIC.

PREMIÈRE PARTIE

Le cahier des charges des SI agiles

L'objectif de cette première partie est d'introduire les problématiques de rationalisation et d'agilité des systèmes d'information, puis de présenter un cahier des charges en regard des attentes et des défis actuels du SI. Ce dernier est envisagé sous un angle métier, puis sous un angle technique.

Le premier chapitre présente l'évolution de l'informatique et les problématiques d'entropie des SI ; il montre que l'accélération globale du monde actuel risque d'accroître fortement ces problèmes si rien n'est fait. Une extension Web de ce chapitre montre que l'outillage (architectures, méthodes, techniques) mis à disposition des DSI jusqu'à présent ne résout que partiellement les problématiques de rationalisation du SI.

Le deuxième chapitre présente les exigences métiers et techniques que l'on peut attendre d'une nouvelle démarche d'organisation et d'intégration du SI.

Les parties suivantes montreront comment SOA répond à ces différentes exigences.

1

Accepter l'entropie des systèmes d'information

Objectifs

Ce chapitre rappelle brièvement l'évolution des technologies informatiques et ses conséquences en terme d'entropie pour le système d'information. Il fait ainsi le constat de la difficulté à maintenir un SI homogène et rationnel.

Il évoque ensuite les enjeux liés à la transformation digitale des SI et la pression qui en découle pour ces derniers, et la nécessité d'évoluer vite, comme le font les géants du Web, afin d'éviter une disruption qui serait fatale à l'entreprise.

1. 1 UNE BRÈVE HISTOIRE DE L'INFORMATIQUE

1.1.1 Les technologies *mainframe* et client/serveur

Dans l'histoire de l'informatique, on a tout d'abord conçu des applications monolithiques, les sites centraux, où toute la logique de persistance, de traitement et de présentation était agrégée dans un ensemble indissociable. Toute l'informatique des entreprises était alors concentrée dans un serveur unique : le *mainframe*, qui constituait le SI.

Avec l'invention du PC sont apparues des applications plus légères en architecture client/serveur. Ces architectures ont proposé de déporter les composants d'interface sur les postes de travail en conservant des serveurs pour la persistance. Elles ont ainsi apporté plus de graphisme et plus de sophistication en termes d'ergonomie.

Le faible coût de ces nouvelles applications a permis leur multiplication dans les entreprises. Ces dernières se sont alors dotées de logiciels transverses de messagerie, de gestion financière, d'aide à la décision, etc. De plus, l'informatique a commencé à être utilisée directement par les métiers et progressivement par tous les profils de l'entreprise.

Ainsi, les systèmes d'information ont grandi rapidement à la suite d'initiatives éparses des différents départements et métiers de l'entreprise. L'absence de gouvernance de cette croissance a souvent abouti à des duplications d'informations, saisies et gérées de manière autonome par ces différents services. Le client/serveur a donc créé l'entropie des SI.

1.1.2 Les technologies Web

À la fin des années 1990, les problématiques de déploiement et de gestion des parcs de PC ont montré les limites des applications client/serveur.

Les technologies du Web sont, par essence, ouvertes, interopérables et aptes à fonctionner sur des réseaux distants. Les années 2000 ont vu leur adoption massive par les entreprises traditionnelles du tertiaire, d'abord pour connecter les clients de l'entreprise, puis les partenaires via les Web Services et l'émergence de SOA. En parallèle, les géants du Web, comme Google, Amazon ou Facebook, ont démontré dans ces mêmes années leurs capacités à gérer d'immenses volumes d'utilisateurs (1,7 milliard pour Facebook à fin 2016) et de données (des centaines de millions de vidéos vues chaque jour sur Youtube).

Ainsi le Web est devenu une plate-forme technologique de référence utilisée dans de nombreux SI. Elle a beaucoup évolué ces dernières années grâce à un riche écosystème de frameworks dans divers langages de programmation (Node.JS, Ruby On Rails, Groovy, Spring Boot...). Ces frameworks permettent un développement rapide grâce à des outils de packaging de composants (npm, bundle, bower...). Ils embarquent des outils de tests unitaires qui garantissent l'usage de bonnes pratiques de « développement piloté par les tests ».

En 2014, HTML5 a transformé la plate-forme Web en une véritable technologie d'interface client/serveur capable de fonctionner en mode déconnecté (web storage, local storage) et en mode asynchrone (web sockets), de créer des objets 2D et 3D (Canvas), etc. Ainsi, l'état de l'art est aujourd'hui la *Single Page Application*, une interface Web événementielle comparable à une application Visual Basic, donc assez éloigné du paradigme de navigation page à page. De fait, les technologies d'interface Web (GWT, Silverlight, Ember.JS, Angular, React, Ionic…) se sont succédé à un rythme rapide et même difficile à suivre par les entreprises. Le chapitre 10 clarifie les aspects architecturaux de ces technologies d'interface.

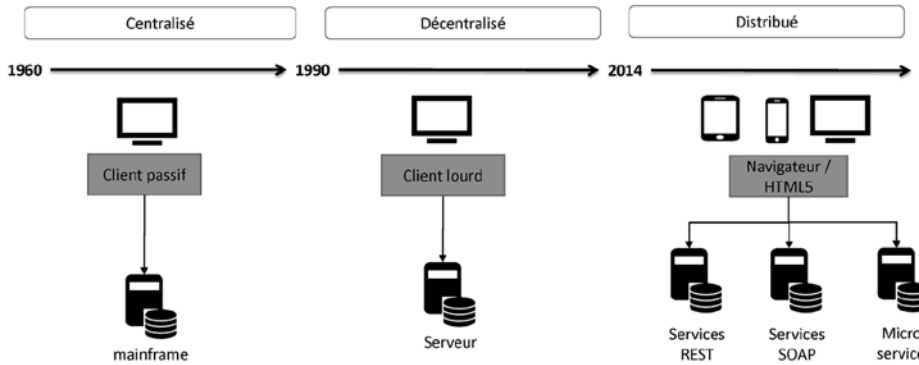
Enfin, les géants du Web ont popularisé la mise à disposition de services comme Google Maps API, reposant sur le style d'architecture REST (partie 6). Ils ont ainsi créé des architectures distribuées avec une très forte capacité technique à monter en charge.

1.1.3 Le *device agnostic*

La sortie de l'iPhone en 2007 puis de l'iPad en 2010 a donné le coup d'envoi des applications mobiles pour smartphones et tablettes. Les géants du Web ont alors commencé à proposer des interfaces *device agnostic*, c'est-à-dire utilisables sur tout

type d'appareil. Ainsi, début 2016, 1,5 milliards d'utilisateurs consultent Facebook sur mobile. Les entreprises comme les banques ou les groupes de presse ont rapidement suivi ce mouvement. Les services exposés sur Internet en architecture REST sont ainsi de plus en plus distribués et consommés depuis des terminaux mobiles.

Figure 1.1 – L'évolution des interfaces homme/machine



De nouvelles générations d'interfaces semblent être amenées à se développer en 2017 :

- ✓ La réalité virtuelle avec des technologies comme Facebook Oculus ou Microsoft Hololens,
- ✓ Les chatbots ou assistants vocaux comme Amazon Alexa, Apple Siri, IBM Watson, Google Assistant ou Microsoft Cortana,
- ✓ Dans une moindre mesure, les appareils *wearable* comme les montres ou les vêtements connectés.

Ces nouvelles interfaces seront amenées à augmenter le nombre de requêtes sur les services.

1.1.4 Le *cloud computing*

Depuis la fin des années 2000, le *cloud computing* est devenu une opportunité intéressante pour les entreprises qui souhaitent externaliser leur informatique. Les bénéfices de ce nouveau modèle sont les suivants :

- ✓ L'infrastructure informatique est vue comme une **commodité** dont on délègue l'exploitation à des spécialistes. Ces spécialistes gèrent des plate-formes de grande envergure, robustes et largement automatisées. Ils parviennent souvent à proposer des tarifs compétitifs grâce à un effet d'échelle.
- ✓ Le *cloud* fournit des **services élastiques**, c'est-à-dire qu'on peut augmenter ou diminuer les ressources utilisées à la demande, et cela à partir d'une simple console Web.
- ✓ Le **paiement suit la consommation de ressources**, donc il n'y a pas d'investissement préalable et le cloud peut absorber des variations de charge sans coût prohibitif.

Les opérateurs *cloud* proposent deux grandes familles de services : des applications clés en main (SaaS : outils collaboratifs, outils de CRM, progiciels de type ERP, etc.) et des plate-formes d'hébergement sous la forme de serveurs d'applications en ligne (PaaS) ou de système d'exécution de serveurs virtuels (IaaS).

Le *cloud* fait émerger des problématiques d'intégration critiques :

- ✓ possibilité d'échanges de données entre le SI et le *cloud* ;
- ✓ sécurisation de ces flux d'échange via Internet ;
- ✓ latence du réseau Internet ;
- ✓ authentification unifiée des utilisateurs accédant à des services internes et externes au SI.

Ainsi le *cloud* contribue largement à l'entropie du SI évoquée plus haut.

1.1.5 De nouvelles architectures qui « passent à la grande échelle »

On a vu plus haut que les géants du Web gèrent des volumes colossaux d'utilisateurs et de données. Leurs contraintes les ont donc amenés à concevoir de nouvelles architectures informatiques, en particulier celles qui sous-tendent le concept du Big Data. Ainsi la plate-forme Apache Hadoop a été créée à la suite d'une publication scientifique de Google sur « Google File System ». Et les technologies de stockage NoSQL[1] qui permettent d'obtenir des temps de réponses rapides en lecture/écriture, en sacrifiant la norme SQL, sont nées chez Amazon, Facebook ou LinkedIn. Enfin, des technologies comme Spark ou Kafka leur permettent d'absorber de gigantesques flots de données[2] en quasi-temps réel.

Par ailleurs, ces mêmes géants du Web opèrent des centres de données de grande envergure (plusieurs millions de serveurs chez Google et Microsoft). Afin de réduire la facture matérielle et énergétique de ces *data centers* et d'automatiser leur production, ils ont été amenés à remettre en question la conception classique des *data centers*. Ainsi Google et Facebook conçoivent eux-mêmes[3] leurs *data centers* et leurs serveurs « du sol au plafond », plutôt que de recourir à des produits Dell ou HP.

Ils conçoivent aussi les logiciels de pilotage des *data centers*, en partant d'une approche originale nommée « design for failure » : le matériel est considéré comme faillible, il est donc conçu pour être peu coûteux (*commodity hardware*), et la résilience est assurée par une couche logicielle qui gère automatiquement les pannes de serveurs.

Pour adresser des architectures à très forte charge, les communautés *open source* ont été amenées à travailler sur des architectures asynchrones : c'est ainsi qu'ont émergé les architectures de type CEP (*complex event processing*) et réactives (chapitre 8).

Ces nouvelles pratiques architecturales deviennent incontournables pour les entreprises amenées à créer des services à forte charge.

---

1. Not only SQL.
2. Les anglo-saxons parlent de « data streaming ».
3. Les résultats de leurs recherches sont disponibles au sein du projet Open Compute.

1.1.6 L'internet des objets

L'émergence de nouvelles typologies d'interface utilisateurs accroît la charge sur les services. Les objets connectés, qui communiquent avec le SI sans interface utilisateur ni même intervention utilisateur, pourraient multiplier fortement cette pression.

En effet, on voit émerger de nombreux cas d'usages intéressants autour de la connexion des objets aux SI traditionnels :

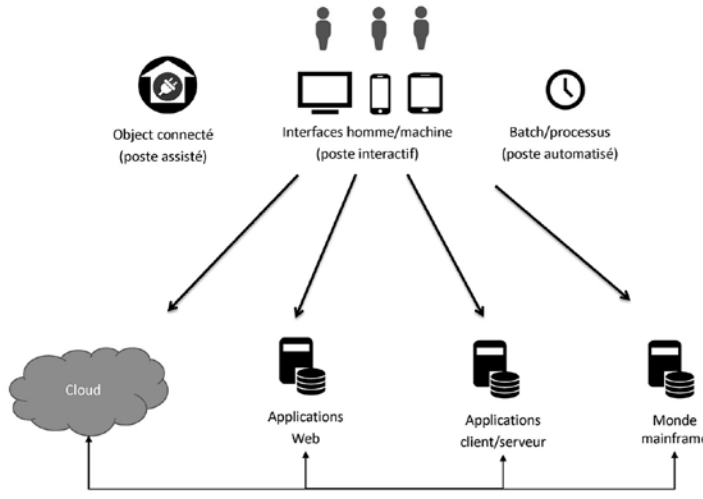
- ✓ **Logement connecté** pour assurer la sécurité contre les fuites ou les intrusions, ou mieux gérer la consommation énergétique.
- ✓ **Véhicule connecté** pour adapter les polices d'assurances aux styles de conduite.
- ✓ **Santé connectée** pour suivre une hygiène de vie ou assister une personne en difficulté.
- ✓ **Smart City** pour optimiser les déplacements de personnes ou la gestion des ressources (éclairage, ordures, eau, etc.).

Ces usages vont générer un nombre colossal de requêtes sur le SI. Par exemple, le futur compteur d'électricité intelligent Linky fera une mesure toutes les 30 minutes dans 30 millions de foyers, ce qui représente 1,44 milliard de mesures par jour.

1.1.7 Un premier bilan

Chacune des technologies citées plus haut a eu sa pertinence à un moment donné dans l'histoire de l'informatique. Par conséquent, aucune entreprise ne peut faire table rase à chaque changement de génération technologique. La conservation de ce biotope hétérogène s'explique aussi par les contraintes organisationnelles et financières qui résulteraient d'un tel changement. Ainsi, les directions informatiques ont dû faire face à des parcs informatiques très morcelés et surtout à des applications peu à même de communiquer ou de partager des informations. Par ailleurs, les nouveaux usages du monde numérique augmentent la pression sur le SI et ses services.

Figure 1.2 – L'entropie du SI



1. 2 LA TRANSFORMATION DIGITALE ET SES CONSÉQUENCES SUR LE SI

1.2.1 Le besoin d'agilité

On connaît les disruptions opérées par les géants du Web dans de nombreux secteurs d'activité :

- ✓ Amazon a obligé des enseignes comme la Fnac ou Darty à se réinventer,
- ✓ Facebook est devenu un canal incontournable d'accès à l'information, presque une nouvelle télévision,
- ✓ Google a secoué le monde de l'édition avec Google Books,
- ✓ Uber a transformé le métier des taxis,
- ✓ les opérateurs télécoms luttent pour ne pas devenir de simples tuyaux, laissant la valeur ajoutée aux géants du Web, etc.

Selon l'expression usitée en 2016, les entreprises veulent faire leur « transformation digitale » pour éviter de se faire « ubériser »[4]. Une idée centrale de cette transformation digitale est d'être capable de livrer de plus en plus vite de nouveaux services et produits finis aux utilisateurs.

Le système d'information est impacté par cette accélération ; dans le cadre de processus de plus en plus informatisés, il est au cœur de ces nouvelles exigences de productivité. Les métiers lui demandent d'être extrêmement réactif aux changements pour aider à mettre sur le marché de nouvelles offres. On parle pour cela d'**agilité** : ce terme désigne la capacité technique du SI à supporter des évolutions rapides.

1.2.2 Le SI à l'heure du digital

Afin de disposer d'un SI *digital-ready*, les entreprises s'inspirent de pratiques des géants du Web. Nous allons en lister quelques-unes :

- ✓ **Une conception centrée sur l'utilisateur** : il est aujourd'hui indispensable d'offrir une expérience utilisateur satisfaisante et fluide. Cela signifie un grand soin accordé à l'ergonomie et aux temps de réponse, donc aux front-ends Web et aux applications mobiles.
- ✓ Les offres et produits doivent être **conçues en collaboration avec l'utilisateur**. La doctrine du *lean start-up*[5] dit qu'il faut sortir des fonctionnalités de manière itérative en vérifiant continuellement la satisfaction utilisateur. Si elle n'est pas au rendez-vous, il faut "pivoter", c'est-à-dire s'adapter immédiatement au feedback utilisateur. Il faut aussi savoir tuer rapidement les fausses bonnes idées, selon le principe du *fail fast*.

---

4. Voir l'interview de Maurice Lévy, patron de Publicis, dans le *Financial Times* en 2014, à l'adresse suivante : https://www.ft.com/content/377f7054-81ef-11e4-b9d0-00144feabdc0

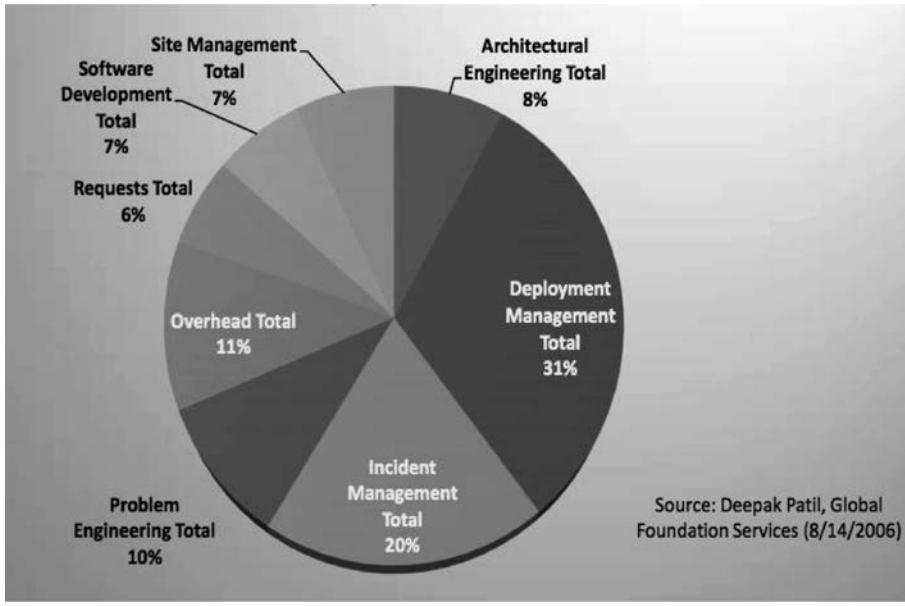
5. *Lean start-up*, Eric Ries, Pearson, 2012.

- ✓ Atteindre un **haut niveau de qualité** dans les développements en s'équipant d'un solide outillage de tests[6] pour éviter les régressions et la dette technique. Cela signifie avoir des développeurs salariés par l'entreprise plutôt que des consultants issus d'ESN[7], afin de maîtriser cet actif stratégique qu'est le code métier.
- ✓ **Reposer sur un modèle de services** comme c'est la règle chez Amazon[8] ou Linkedin.
- ✓ Être capable de **déployer les services en continu**, éventuellement toutes les semaines, afin de ne pas se laisser distancer sur le marché. Cela repose sur des pratiques DevOps (voir encart ci-dessous) et des architectures qui facilitent ce déploiement comme les plate-formes Cloud et les microservices[9].

La mise en œuvre rapide de toutes ces pratiques est très complexe pour une entreprise qui dispose d'un fort existant informatique et d'habitudes de travail ancrées.

Pour aller progressivement vers cette cible, le Gartner parle de « bi-modal IT ». Il s'agit d'appliquer les pratiques des grands acteurs du Web pour les projets innovants, et de conserver les anciennes pratiques pour les systèmes moins récents comme le mainframe. On verra qu'il est possible d'exposer des services à partir des données du mainframe pour permettre aux projets innovants d'en tirer le meilleur parti.

Figure 1.3 – Le quotidien des équipes OPS



6. Tests unitaires, tests d'acceptance, tests de performance, etc.

7. Entreprises de services du numérique

8. Voir le Memo de 2012 de Jeff Bezos : "All teams will henceforth expose their data and functionality through service interfaces (...) Anyone who doesn't do this will be fired". Cf. https://apievangelist.com/2012/01/12/the-secret-to-amazons-success-internal-apis/

9. Les microservices sont introduits au chapitre 3.

DevOps

Le concept de DevOps est issu de la volonté de briser la frontière entre :

- **les développeurs** (les Dev) dont l'objectif premier est de livrer des nouveautés, éventuellement en introduisant des technologies émergentes,
- **la production** (les Ops) dont le but est avant tout de garantir la stabilité de l'infrastructure grâce à des technologies maîtrisées.

Il s'agit d'essayer de résoudre une tension entre des objectifs locaux qui peuvent sembler opposés, même si, au final, tout le monde œuvre pour les utilisateurs de l'entreprise. En outre, une étude de 2006[10] a montré que 51 % du temps total des activités Ops est consacré au déploiement applicatif, souvent parce que les Dev leur remettent du code impropre à la mise en production. Une meilleure collaboration apparaît donc indispensable.

Pour améliorer cette collaboration, DevOps propose trois ensembles de bonnes pratiques (voir aussi le chapitre 12).

- Les Devs partagent leur outillage d'industrialisation des développements en proposant aux Ops le « **continuous delivery** », c'est-à-dire un outillage sophistiqué de déploiement[11] du code produit permettant la mise en production automatique. Les mises en ligne sont donc fréquentes, parfois, plusieurs fois par jour chez certains géants du Web. Au premier abord, ces mises en ligne fréquentes peuvent paraître risquées. En pratique, on constate qu'il est plus aisé de déployer quelques lignes de code qu'une mise à jour majeure. Le risque de problème (régression, corruption de données, etc.) est moins important, le retour en arrière plus facile. Il est d'autant plus facile qu'on aura adopté une stratégie de mise en conteneur avec un outil comme Docker (présenté aux chapitres 4 et 19). Par ailleurs, lorsque le processus de déploiement devient habituel et banalisé, il est beaucoup mieux maîtrisé : lorsque l'on fait une mise en production une fois par an, c'est un moment de grand stress et l'on a parfois oublié comment on avait résolu les problèmes rencontrés l'année passée.
- Les Ops partagent leur outillage de gestion de plate-forme en proposant aux Devs « **l'infrastructure as code** », c'est-à-dire la possibilité de déployer sur la plate-forme via des scripts automatiques[12] sans intervention manuelle. « Infrastructure as code » permet d'accélérer les phases d'approvisionnement et de mise à disposition des environnements. Un avantage majeur de ces scripts est de rendre les opérations de mise en production reproductibles et testables. Il est ainsi possible de déployer 10 machines virtuelles ou cinq conteneurs (cf. chapitre 19 à propos de Docker) de manière totalement maîtrisée. « Infrastructure as code » est un prérequis au *continous delivery*.

10. Étude de Deepak Patil (Microsoft Global Foundation Services) de 2006, via une présentation de James Hamilton (Amazon Web Services).

11. Via des outils comme Jenkins ou des plate-formes PaaS comme CloudBees ou Semaphore.

12. Via des outils comme puppet, chef, capistrano, cfengine, ansible...

Les deux équipes **partagent des rituels** réguliers comme des décisions prises en commun sur l'architecture, des revues sur les métriques de la plate-forme, des revues "post mortem" suite à des incidents… Chez Amazon, on dit même « you build it, you run it », c'est-à-dire que les Devs sont impliqués dans les opérations. L'idée sous-jacente est simple : être réveillé à 3 heures du matin parce qu'une mise en production se passe particulièrement mal est très formateur et sensibilise les développeurs aux problématiques Ops. Inversement, les Ops sont invités à se sensibiliser aux méthodes et outils utilisés par les Dev, pour diminuer l'aversion aux risques, naturelle dans une direction de la production "classique".

DevOps constitue l'intégration des Ops dans une démarche agile. Il facilite les mises en production régulières, donc le fonctionnement itératif. Il pousse les Ops à recourir aux tests, comme le font les Devs.

L'évolution de l'informatique a conduit à des SI hétérogènes et difficiles à maintenir. Par ailleurs, la multiplication des interfaces et des objets connectés augmente la pression sur le SI, l'intégration de plus en plus forte entre systèmes internes et externes poussent à considérer le SI de plus en plus distribué, et la transformation digitale le force à évoluer rapidement. Il devient donc vital de trouver des moyens de faciliter les transitions continues du SI.

Définir le cahier des charges du style SOA

Objectifs

Empiler les outils ne suffit pas pour rationaliser le SI ou moderniser son architecture. La démarche d'urbanisation souhaite parvenir à aligner les besoins métiers et l'architecture SI. Cependant, le patrimoine informatique est conséquent et il faut composer avec. Il convient donc d'engager une transformation pas à pas du SI.

Le virage vers le digital amène son lot d'exigences nouvelles pour le SI et une plus large ouverture de tous les canaux de communication de l'entreprise. Ceci s'accompagne d'une révision profonde du modèle de fonctionnement du SI, que ce soit sur le plan organisationnel, opérationnel, ou technique.

L'objectif de ce chapitre est donc double : d'une part, cerner les besoins complémentaires métiers et techniques qui nourrissent le recours à un nouveau style de construction du SI, d'autre part, il s'agit de préciser :

- quelles fonctions devraient offrir un SI pour mieux répondre aux besoins des métiers de l'entreprise,

- quelles fonctions devraient offrir les socles techniques d'un SI pour satisfaire les urbanistes et la DSI.

Le « style SOA » va peu à peu cristalliser les réponses architecturales à ce nouveau modèle. Ce chapitre résume l'organisation logique des thèmes de cet ouvrage pour aborder chacun des principaux défis liés à une approche SOA.

2. 1 LES BESOINS DES MÉTIERS

L'informatique est omniprésente voire critique pour le développement et la survie de l'entreprise. Elle doit suivre en continu les besoins métiers et stratégiques. L'évolution des usages fait naître d'autres exigences abordées ici.

2.1.1 Soigner l'expérience client derrière chaque produit digital

Le succès des géants du Web, des applications mobiles et de la majorité des logiciels majeurs le confirme régulièrement : il est impératif de soigner son produit et sa prestation pour les rendre plus utiles, plus faciles d'usage, plus efficaces, plus accessibles, plus crédibles et même plus attrayants que d'autres. L'émergence des objets connectés renforce ce besoin pour le **produit digital** qui mixe désormais l'objet physique[1] et le logiciel[2].

Cette expérience d'utilisateur, en particulier d'un client ou d'un abonné, doit être au cœur de la réflexion d'une prestation. Dans le monde digital, le client veut à la fois :

- ✓ obtenir l'information qu'il souhaite en seulement quelques clics,
- ✓ pouvoir installer, essayer et utiliser son produit très rapidement.

La concurrence fait rage pour délivrer le meilleur service au client. Mais chaque entreprise doit-elle se transformer en éditeur de logiciels ou revendeur d'objets connectés ? La DSI, à elle seule, n'est pas toujours en mesure de surmonter ces difficultés. Faut-il délivrer un seul très bon produit, ou au contraire en lancer plusieurs en même temps pour laisser la sélection darwinienne opérer ?

L'entreprise doit en amont clarifier sa stratégie : elle doit reconnaître la meilleure chaîne de valeurs pour chaque information clé réclamée par ses clients et qu'elle est (seule) capable de délivrer. Rares sont les cas où le client peut se satisfaire des données brutes maintenues dans le SI. Celles-ci doivent être soignées et transformées en information pour prendre de la valeur. Les efforts doivent se concentrer sur quelques maillons qui représentent la plus forte valeur ajoutée et le véritable cœur métier ou sa cible.

Pour chaque bataille retenue, une unité organisationnelle doit se constituer en équipe produit : le métier, la technique et les représentants d'utilisateurs. Progressivement, l'entreprise doit maîtriser le coût et la valeur de chaque îlot de ressources et peut décider d'en faire un micro-produit différenciant ou pas de son offre. En lien direct avec les différents segments de clients visés, cette recherche induit toujours beaucoup de tests et d'essais à blanc. Elle induit donc un impact IT fort.

Une nouvelle culture du changement et de l'amélioration en continu doit s'établir. Chaque proposition de valeurs doit être soignée pour faire la différence. Il faut itérer pour trouver la meilleure recette.

1. Par exemple le boîtier d'un assureur placé dans un véhicule d'assuré qui permet (1) de moduler la prime d'assurance en fonction du km et de l'entretien et (2) signaler une avarie.

2. Par exemple l'application smartphone qui accompagne le boitier d'un assureur et permet (3) à l'assuré de consulter sa prime réactualisée périodiquement via les informations du boîtier.

L'équipe « produit » peut retenir de piocher davantage à l'extérieur certaines fonctionnalités qui peuvent lui permettre de rester plus performante dans l'expérience qu'elle délivre au global, y compris sur le court terme. Par exemple, en rendant le logiciel indépendant du matériel ou des objets connectés qui s'y rattache, ou en réutilisant des services tiers déjà très répandus (authentification, géolocalisation, etc.). Elle peut aussi proposer des variantes freemium et premium[3] des produits.

Ces améliorations d'expérience utilisateur imposent d'optimiser la chaîne des transactions de bout en bout. Il faut plus de souplesse au moment de changer l'implémentation d'un maillon lorsque c'est nécessaire, notamment ce qui est visible du client. L'organisation doit pour cela établir un cadre de description et une gestion plus systématisée : mesure du coût et de la performance mais aussi de l'usage et de la valeur de son ensemble de logiciels et matériels érigés en produits digitaux. Pour limiter l'effet « mikado » sur le SI, où le bousculement d'un produit peut risquer de bousculer tous les autres, il faut aussi préserver l'autonomie de chaque produit.

2.1.2 Une vision temps réel sur l'activité et l'usage

Pour faire face à l'accélération des marchés et réagir plus efficacement aux points de blocage dans l'expérience client, il ne s'agit plus seulement de consolider des coûts financiers. Les métiers doivent aujourd'hui disposer d'indicateurs en temps réel sur l'activité opérationnelle de l'entreprise autant que sur l'usage client effectif des produits proposés :

- ✓ Accéder aux indicateurs clés sur un client donné : localisations, contrats et chiffre d'affaires, historique de commandes, fiches de satisfaction, etc.
- ✓ Disposer d'indicateurs au fil de l'eau d'un produit donné : commandes en cours, facturation, stocks et autorisations, capacité d'approvisionnement, localisation du produit en cours de livraison, états des capteurs dans le produit, etc.
- ✓ Disposer d'indicateurs comportementaux (saisonnalité, répétabilité, etc.) et d'efficacité sur les processus métier, qu'ils soient fortement ou très peu automatisés.

Pour satisfaire ce type d'exigences, le SI doit permettre de consolider chaque trace d'activité sur toutes les étapes de processus ainsi que les accès aux sources d'information de l'entreprise, et ceci quel que soit le point d'entrée.

Il s'agit de décloisonner des systèmes monolithiques isolés et faciliter la circulation d'informations au sein du SI. Celui-ci devient agnostique sur le plan technologique, c'est-à-dire qu'il ne doit pas rencontrer de point de blocage lié à une hétérogénéité technologique. On verra dès la partie 3 que le style SOA propose différents canevas de construction (« façade », « étagement », « démultiplication », etc.). Il garantit la consolidation de traces et d'indicateurs, tant pour le fournisseur que le client. Il

3. Terminologie anglo-saxonnes pour désigner une offre gratuite en regard d'une offre payante.

les organise au fil des mutations de chaque information « racine » (voir chapitre 7), à mesure qu'elles augmentent, du fait notamment de flots digitaux continus.

2.1.3 Justifier son budget, voire se justifier

Le temps où l'informatique était perçue par les directions et métiers comme une mécanique mystérieuse et impénétrable est révolu. Les DSI ne disposent plus comme par le passé de budgets pour investir sur une nouvelle technologie sans justification ni calcul de retour sur investissement. L'entreprise peut imposer certains choix pour sa survie stratégique dépassant cadre financier ou logique DSI.

L'informatique est aujourd'hui une source d'énergie pour l'entreprise au même titre que l'électricité. Ses actions doivent donc être mesurables, justifiées et rationalisées, et les DSI doivent travailler sur la pérennisation de leurs investissements. De plus, tout nouvel investissement doit être pensé pour servir la stratégie métier et apporter de la valeur.

On ne déclenche plus de Big Bang pour redévelopper entièrement un existant dans le seul objectif d'utiliser la technologie la plus actuelle. On privilégie plutôt l'intégration de la nouveauté dans l'existant.

Dès lors, la DSI se retrouve parfois en concurrence avec la direction marketing souvent devenue spécialiste de l'expérience digitale ou du dernier cri technologique, du prototypage rapide, de la conception de bout en bout, des tests utilisateurs ou des tutoriels interactifs.

Loin des stériles batailles de pouvoirs, l'entreprise et ses directions doivent établir un modèle de proposition de valeurs plus coopératif. Il faut optimiser le coût global d'acquisition, l'intégration et le maintien opérationnel de chaque capacité métier, *a fortiori* lorsque celle-ci est informatisée voire fortement partagée.

Ceci s'applique tant sur le plan métier que sur le plan informatique. Il s'agit d'appréhender différents scénarios d'analyse de la valeur en regard du délai, des coûts, des performances, de la qualité, des compétences nécessaires, de l'utilité véritable d'un savoir-faire distinctif, etc. Ce modèle s'étend de proche en proche jusqu'aux partenaires, au moins pour ceux considérés comme stratégiques. Il devient alors possible d'adapter le bon niveau de synergie entre des initiatives pouvant naître localement ou en central pour qu'elles puissent s'étendre plus globalement.

On verra dès la partie 3, puis en partie 4 et 6 comment le style SOA s'envisage à l'échelle globale de l'entreprise. Il répond à cette exigence de suivi de la rentabilité et du bon usage d'un investissement informatique pour disposer de produits répondant aux attentes client sur chaque domaine d'activité. Ceci facilite une transition progressive de l'entreprise vers un modèle capacitaire plus modulaire. Ces « modules » de capacités métier nécessaires à chaque produit fini, évoluent selon un arbitrage des moyens et ressources pour le bénéfice global de l'entreprise.

2.1.4 Adapter la digitalisation des processus métiers aux réalités du terrain

L'entreprise cherche trop souvent à informatiser rapidement tous ses processus métiers. L'erreur a longtemps été de vouloir imiter le processus manuel, historiquement lié à une délégation hiérarchique des tâches. Mais on doit constater que tous les processus métiers ne se ressemblent pas, et la plupart sont suffisamment complexes pour qu'il devienne irréaliste de les automatiser entièrement. Leur spectre très large fait qu'ils sont plus ou moins répétables, par exemple, dans le contrôle réglementaire, la commande de produits ou dans certaines transactions financières. Ils peuvent aussi être beaucoup plus ad-hoc, voire non déterministes, comme dans l'innovation, l'investigation policière ou le soin d'un patient par exemple.

Il ne s'agit plus de croire en l'outil miracle de *workflow* pour apporter la réponse informatique à cette diversité de métier. Dans le même temps, la diversité des échanges opérés durant ces processus, continue, elle, de s'étendre au sein de l'entreprise comme à l'extérieur, impliquant plusieurs réseaux de participants à la fois humains, logiciels, automates et même objets connectés.

L'entreprise doit désormais proposer des solutions adaptables à ce large spectre des typologies de processus métier mais aussi définir et construire l'architecture permettant d'y intégrer un réseau étendu de participants.

Il sera plus efficace d'automatiser une procédure structurée lorsqu'il est possible d'en établir un modèle relativement stable et que son exécution se révèle répétitive. En revanche, il est plus efficient d'assister un participant humain dans une procédure *ad hoc*, sans lui imposer un comportement *a priori* dicté par la machine : en suggérant, vérifiant ou dématérialisant plus aisément certaines tâches qu'il doit effectuer.

Les processus métiers impliquant directement le client final requièrent un dossier d'informations souvent éparpillées bien au-delà du seul contexte du processus. Il faut réunir et compiler quasiment en temps réel toutes les connaissances utiles dans une étape donnée du processus afin de prendre la meilleure décision quels que soient les canaux de provenance : décision hiérarchique, signaux transmis par un objet connecté, alerte ou flots de réactions dans des réseaux sociaux, etc.

Les outils doivent donc être plus souples et plus larges dans leur spectre de processus que les outils de *workflow*. On verra en partie 3 (chapitre 9) que le style SOA intègre la prise en compte d'une diversité de participants acteurs de processus métiers complexes. Ceci, en facilitant l'automatisation de processus transverses lorsque cela se révèle efficace et efficient à travers plusieurs silos[4], mais aussi en facilitant l'assistance de l'humain dans une boucle de décision adaptée pour un contexte complexe ou inhabituel.

4. Voir plus haut le besoin d'indicateurs et le suivi d'activités impliquant plusieurs participants.

2.1.5 Protéger l'information, un incontournable

L'influence de la globalisation des marchés et des réglementations nationales voire internationales pousse nombre d'organisations à d'énormes investissements de remise en conformité, notamment sur la sécurité de leurs données. Hier encore tenues secrètes et très individualisées, ces politiques doivent désormais se conformer à des règles de transparence. La difficulté se trouve accentuée d'une part du fait de l'éclatement en silos dans l'entreprise, mais aussi du fait du renouvellement de ces directives réglementaires. Le travail de l'urbaniste permet de cerner les blocs de capacités métier concernés et clarifier les impacts de chaque article de réglementation. La DSI doit dans ce cas rester force de proposition pour établir un cadre de formalisation et de contrôle transversal de ces directives.

Le problème est complexe puisqu'au sein même d'une seule unité organisationnelle des rôles et des accès différents peuvent s'établir par rapport à un portefeuille d'informations réparties ou des activités localisées (ex : les employés commerciaux d'antennes régionales ne doivent pas consulter ou signer des dossiers non relatifs à leur antenne ou au-dessus d'un certain montant d'argent, sans consentement du directeur de cette antenne). La possibilité d'envisager un seul langage d'expression de n'importe quelle directive métier est un mythe. Il est en revanche possible de graduer la formalisation d'une identité numérique jusqu'à un cadre légal individuel[5] afin de rendre non répudiables certaines transactions sensibles dans le monde digital (signature d'actes authentiques, etc.).

Dans ce contexte de transparence, il n'est plus possible d'accéder à l'information sauvagement ou de manière directe par chaque application qui en a besoin : une voie d'accès doit lui être ouverte au préalable.

On verra en partie 2 et 6 que le style SOA insuffle un modèle de formalisation et de vérification de directives de sécurité sur les opérations les plus sensibles pour chaque typologie de clients. On parle à ce titre de contrat d'accès dans l'orientation service. Ce modèle s'établit au sein d'une façade d'accès à ces opérations. Reste alors à se prémunir d'accès court-circuitant ces façades.

2.1.6 L'agilité, un objectif global de construction pour l'entreprise

Le contexte économique contraint l'entreprise au réajustement organisationnel quasi permanent. L'entreprise requiert dès lors un recours simplifié et modulable à toutes les capacités métier pour y effectuer des réajustements et en particulier arbitrer plus ou moins fréquemment entre :

- ✓ la réutilisation du patrimoine informatique, quitte à l'adapter ou le compléter,
- ✓ la réalisation d'une (nouvelle) implémentation ou brique technologique,

5. À ce jour, au moins pour les personnes physiques et selon la législation locale.

✓ l'acquisition d'un outillage du marché à déployer en interne ou via le Cloud.

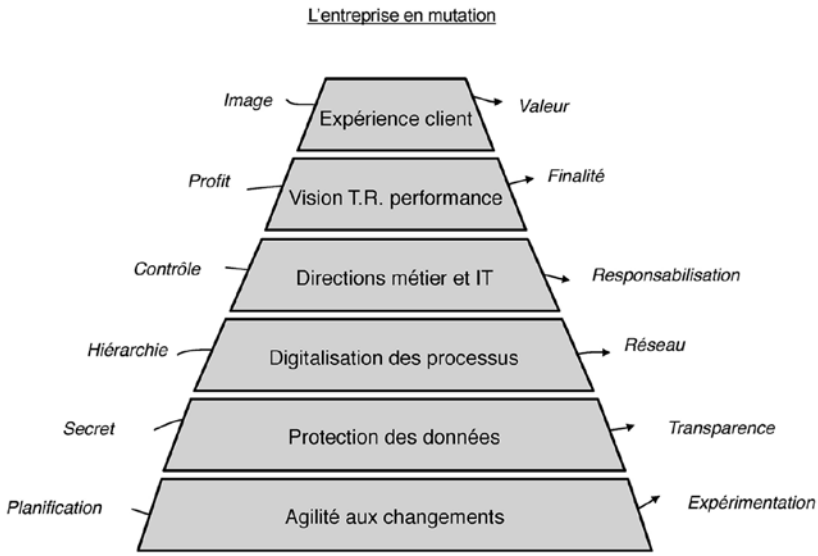
Dans le même temps, l'entreprise est aussi en innovation permanente pour trouver de nouvelles niches de marché ou satisfaire à des nouveaux usages et établir l'expérience client la plus adaptée. Elle construit alors des initiatives indépendantes.

> L'entreprise recherche désormais un cycle continu pour moduler et adapter ses capacités (activités et ressources clefs) selon les objectifs de sa stratégie et les facteurs de changements de son environnement. Le fruit de plus fréquentes expérimentations devient source d'amélioration.

Dans ces ajustements, il convient de pouvoir rapidement arrêter ou renforcer une capacité métier ou technique et optimiser son modèle économique, qu'il soit existant ou émergeant. Par exemple, transformer certaines capacités en commodités pour accélérer davantage la délivrance du produit ou service apporté, ou encore abandonner certains pans d'activités sans entraver ceux qui fonctionnent. Derrière cette apparence d'instabilité se cache la nécessité de conduire chaque transformation de manière plus structurée et maîtrisée : converger vers une cible souhaitée et non au chaos. On verra dès la partie 2 que le style SOA permet de garder le focus sur l'adaptation nécessaire et suffisante à la diversité de clients envisagés[6]. Il isole ces derniers de la transformation des capacités nécessaire au fil de plusieurs transitions d'implémentation d'un métier.

La figure 2.1 résume les exigences métiers d'un nouveau style de construction du SI.

Figure 2.1 – La vue d'ensemble du cahier des charges métiers



6. Les anglo-saxons parlent d'approche « outside-in ».

2. 2 LES EXIGENCES TECHNIQUES

D'une manière générale, la DSI bascule d'une culture de moyens à une culture de résultats. Elle doit remplir le cahier des charges métiers au meilleur « time-to-market ». Les chantiers entrepris doivent s'inscrire dans la lignée des objectifs d'apport de valeur et d'amélioration de performance fixés par l'entreprise. Sur le plan architectural, on peut considérer deux volets majeurs d'exigences :

- ✓ L'ouverture de frontaux du SI et le virage digital pour toutes les cibles de l'entreprise : clients particuliers et professionnels, partenaires, employés ;
- ✓ L'allégement des monolithes et infrastructures lourdes du SI pour soutenir l'agilité requise.

Ces exigences de transformations techniques s'accompagnent aussi pour la DSI d'un troisième volet d'exigences organisationnelles pour instaurer des transitions stables et des itérations sur le rythme du métier.

En filigrane à ces exigences apparaît un changement profond du mode de construction du SI. Il est nécessaire de maîtriser sa complexité en le découpant en morceaux autonomes, connectés et réutilisables si besoin : c'est là le but premier du style SOA.

2.2.1 Le *front office*, premier volet d'exigences techniques

◆ *Ouvrir ma vitrine, ma boutique, mon guichet, mon réseau social : maintenant !*

L'Internet grand public a montré la voie des possibles pour l'entreprise. Celle-ci exige désormais de disposer de **fonctions prêtes à l'emploi** pour élargir ses canaux traditionnels d'activités. Elle réclame implicitement de disposer d'un libre-service constituant son **minimum vital en ligne** : l'inscription d'utilisateurs, la personnalisation d'espaces de relations ou de collaborations, le dépôt de commandes ou la formulation de demandes et de réclamations, le catalogue des offres et le paiement de produits et prestations. Et la direction bien souvent insiste sur ce point : « il le faut pour hier et disponible sur le Web comme sur mobiles ! ».

La DSI ne dispose que d'implémentations ou d'outillages couvrant partiellement ces exigences. Cet ensemble « frontal » est généralement fortement contraint en termes de **disponibilité, de robustesse et de performance**. Il faut que cela tienne même lors d'interruptions de l'ERP ou du mainframe central. Le « frontal » voit aussi sa **présentation**[7] **revue fréquemment**. Ceci est lié à l'évolution de la charte graphique et l'expérience utilisateur que l'entreprise choisit selon sa communication vis-à-vis de segments de marchés. Enfin, chaque fonction doit **s'intégrer avec l'existant**

7. Les anglo-saxons parlent de « look and feel ».

des systèmes patrimoniaux, détenteurs des données utiles. L'entreprise ne peut se résoudre à ne satisfaire qu'une partie de ces besoins. Pour aller plus vite et éviter de réinventer la roue, la DSI peut choisir d'intégrer certaines fonctionnalités manquantes à travers des partenaires **tiers sur le cloud**[8]**.**

On verra dès la partie 2 que le style SOA permet d'abstraire les implémentations et la présentation de fonctions transverses telles qu'énumérées ici au travers d'interfaces programmables (ou API). La DSI et le métier doivent adapter, publier, sécuriser et maintenir ces API vis-à-vis de leurs clients. Une concurrence à la meilleure implémentation peut itérativement améliorer ce « frontal ». La partie 5 aborde en détail les exigences liées à la robustesse, la performance et la distribution de telles architectures.

◆ *Rester connecté*

L'ère digitale instaure l'avancée de terminaux **mobiles**, l'essor des objets connectés et de multiples autres équipements de plus en plus surdensifiés en capteurs. Le **Web** et **l'Internet des objets** (IoT) sont devenus des incontournables pour des secteurs de plus en plus nombreux : usines de fabrication, transporteurs, établissements médicaux, centrales et transports d'énergies, aéroports, villes et services d'administration publique, voitures et maisons connectées, etc.

Le SI doit être de plus en plus **réactif** face à la banalisation progressive de terminaux diversifiés. Cette réactivité doit être en quasi temps réel dans la mesure où le système d'information est en interaction non seulement avec des êtres humains, mais aussi, et de plus en plus, avec des objets et robots connectés, engagés dans divers types **d'interactions de « machines à machines »**. Cette évolution a un impact fort sur le SI patrimonial et toute sa périphérie.

D'année en année, la puissance et la distribution des capteurs et micro-terminaux multiplient la densité des volumes et les contraintes de réactivité, de sécurité, de robustesse et de performance pour traiter des micro-événements.

Cette masse d'information ne doit pas noyer l'utilisateur (ni faire écrouler le SI !). La DSI doit aussi apporter des réponses pour capturer, sécuriser, canaliser, échelonner, filtrer, stocker, diffuser ces millions de micro-événements.

Dans cet environnement ultra-connecté, le passage en **mode déconnecté** est un événement attendu à concilier avec la cohérence et la disponibilité des données. On verra par la suite que du point de vue architectural, il peut s'apparenter à un partitionnement entre deux sous-réseaux (chapitre 15), similaire à des événements – moins attendus – causés par une véritable panne réseau.

8. Par exemple Amazon Web Services IaaS, Magento Commerce PaaS, etc.

L'invasion de protocoles optimisés

Dans ce raz de marée technologique, apparaissent en bout de chaîne de nouveaux **protocoles** d'échanges de messages **spécialisés**. Ils font dialoguer ces terminaux ou objets communiquant avec des passerelles plus classiques du SI sur le monde IP[9], et des échanges directs et optimisés entre des technologies hétérogènes. Citons quelques exemples avec :

- des mobiles (protocoles mobiles : OTA, OMA…),
- des objets connectés (protocoles LPWAN : Sigfox, Lora...),
- des logiciels distants (protocoles sérialisant : ProtoBuf, Thrift, gRPC…).

On verra dès la partie 2 comment le style SOA facilite l'intégration de cet écosystème digital en périphérie du SI. Ceci peut nécessiter le recours à de multiples socles techniques : pour la gestion de parc d'équipements mobile (Mobile Device Management), la gestion d'objets connectés et de passerelles avec des objets connectés (Fog Computing)[10], pour la concentration des flots de micro-événements et leur consolidation en événements de plus gros grain, plus directement utilisables pour le SI de gestion ou pour des acteurs humains.

2.2.2 Le *back-office*, second volet d'exigences techniques

Une fois les facteurs d'entropie du SI admis, la DSI se doit d'établir et faire respecter un nouveau style de construction et d'intégration nécessaire aussi bien dans les solutions métiers qu'elle délivre, que dans les progiciels et socles techniques qu'elle choisit plus transversalement. Il s'agit donc de s'attaquer à ses « paquebots » applicatifs : les monolithes qui supportent aujourd'hui encore l'essentiel du cœur métier de l'entreprise.

◆ ***Passer du monolithe à des solutions modulaires***

Le SI est complexe et distribué. Ce constat fait travailler ensemble DSI et métier au découpage de justes frontières au sein du patrimoine informatique.

L'exercice revient à imaginer « comment découper mon actuel monolithe ou mon SI, dans un monde idéal, sachant à l'avance tout ce qu'il fait maintenant et doit encore faire prochainement ». Les urbanistes définissent une cible de blocs fonctionnels autonomes (ou modules) jusque dans l'infrastructure opérationnelle. On planifie l'encapsulation progressive de l'existant dans ces blocs. Il faut découpler peu à peu l'ensemble complexe actuel de logiciels puis intégrer plusieurs modules pour constituer des solutions métiers adaptées.

9. Par exemple HTTP, JRMP, COM+.

10. Voir plus loin d'autres exigences sur les socles d'infrastructure.

Outre la possibilité de s'affranchir de choix technologiques, on verra que cette approche permet d'exécuter des transitions de transformation d'une architecture existante en facilitant l'ajout ou le retrait de capacités dans le SI ou sur le cloud.

◆ *Manipuler des informations de plus en plus distribuées*

Dans l'ère digitale, le recours aux objets connectés et le besoin de consolidations massives d'informations réparties entre de multiples applications internes ou externes, conduit à l'augmentation des entrepôts et des volumes de données autour du SI. Dans le même temps demeure la nécessité de manipuler ces données distribuées et redondées sur plusieurs nœuds. Qu'il s'agisse de prendre en compte le vote d'électeurs, suivre le comportement de l'usager d'un portail, utiliser les mesures de capteurs embarqués, le SI peut vite se retrouver bombardé de données à absorber, enregistrer, traiter, consolider et diffuser dans les applications. L'infrastructure physique du traitement de ces flots de données nécessite beaucoup de ressources machines : le travail de l'architecte du SI est désormais d'organiser et structurer les traitements et les échanges sur ces données.

On verra dans la partie 3, que le style SOA répond à ces besoins, d'une part via des socles techniques pour concentrer et étager des flots de données selon l'intensité de leur débit, et d'autre part en proposant des canevas de construction éprouvés pour bâtir les composants en charge de l'exécution d'une logique métier. Il s'agit de savoir arbitrer chaque construction selon des contraintes particulières de robustesse et de performance.

◆ *Alléger les socles d'outils*

Cette démarche de construction en blocs modulaires peut réclamer le recours à différents socles d'outillage. Le style SOA n'échappe pas à la règle. On distingue ici quatre grandes familles de socles pour :

- ✓ L'exécution opérationnelle de composants, on parle de *socles d'exécution*,
- ✓ Les communications entre composants requièrent des *socles middlewares*,
- ✓ La configuration et l'exploitation requièrent des *socles gestionnaires*.
- ✓ Les développements logiciels, on parle de *frameworks* et *fabrique*,

Le discours des éditeurs a tendance à faire croire que ces différents socles peuvent être centralisés au sein d'une seule plate-forme, devenant alors un point sensible de défaillances.

Une bonne délimitation des frontières entre les différentes capacités techniques et la chaîne d'intégration requise selon les systèmes en présence, permet de cibler le recours aux outils appropriés ainsi que leur empreinte technologique plus ou moins structurante pour le SI.

◆ *Intégrer la juste chaîne de commodités techniques*

De manière générale, on distingue les principales capacités techniques suivantes :

La connectivité (ou adaptateurs)

Cette capacité couvre la gestion du dialogue technique à bas niveau vers une technologie propriétaire d'une ou plusieurs applications du SI. Selon les librairies ou API mises à disposition par le fournisseur de chaque application, le connecteur fait usage d'un protocole plus ou moins riche (FTP, SQL*Net, RFC, EDIFact, HL7, etc.) pour communiquer avec l'application. Certains connecteurs peuvent notamment assurer des contraintes de journalisation des accès, de transactionnalité lors d'accès en écriture, de lectures ou écritures en masse, ou encore d'un formatage syntaxique des données (binaire, CSV, XML, JSON, etc.).

L'acheminement de messages à « gros grain » (ou flux)

Cette capacité couvre la gestion du transport et de files d'attente pour des échanges par messages entre plusieurs logiciels. Ces messages peuvent correspondre à des informations circulant entre différents référentiels, des transactions à réaliser entre des applications, ou encore des événements entre plusieurs processus métiers.

L'acheminement de messages à « grain fin » (ou flots)

Cette capacité couvre la gestion du transport et de la répartition de messages porteurs d'événements temporels (clic souris, températures d'un capteur, etc.). Il s'agit d'échanges d'une courte information, mais sur un débit ou un nombre généralement très élevé, notamment entre le SI et des passerelles vers le monde digital (Cloud, objets connectés, etc.).

La médiation

Cette capacité permet de transformer les informations entre logiciels. Ceci peut s'effectuer par l'ordonnancement de traitements par lots (ou batchs), la gestion de procédures techniques d'extraction ou d'injection de données en masse, ou encore la gestion de règles de validation, enrichissement, transformation, routage ou corrélation lors d'échanges de messages entre ces logiciels.

La coordination

Cette capacité couvre la gestion de conversations entre les applications. Celles-ci peuvent être figées à la construction (on parle de composition). Elles peuvent aussi être dirigées par une application qualifiée alors d'« orchestrateur » principal du dialogue avec toutes les autres, ou encore faire l'objet d'une convention établie entre plusieurs applications (on parle de chorégraphie), ou même être annoncées dynamiquement au fur et à mesure de cette conversation (conversations dites HATEOAS[11] sur HTTP)

La contractualisation

Cette capacité couvre l'enregistrement et l'interrogation de descripteurs et de références uniques sur des messages métiers ou techniques. En effet, chaque pourvoyeur de messages peut être physiquement instancié ou distribué sur plusieurs points d'accès pour des contraintes de robustesse ou de performance. Les applications clientes

11. HATEOAS est l'acronyme anglo-saxons de « Hypermedia as the Engine of Application State ». Il annonce les conversations possibles par des liens hypertextes.

doivent notamment connaître ces descripteurs afin de fabriquer les messages adéquats d'une instance valide du pourvoyeur. On peut selon les cas, souhaiter contrôler des directives particulières par message (conditions de restriction du trafic, de sécurisation de l'échange, garantie de délivrance du message, etc.).

L'exposition

Cette capacité couvre la gestion de descripteurs d'interfaces programmatiques (API) sur des prestations érigées en produit à l'échelle du SI. Il s'agit ici de promouvoir un accès soigné[12] aux populations de développeurs clients du SI, qu'ils soient internes ou externes à l'entreprise.

Enfin, deux capacités se retrouvent transversalement en soutien et en pilotage :

La construction

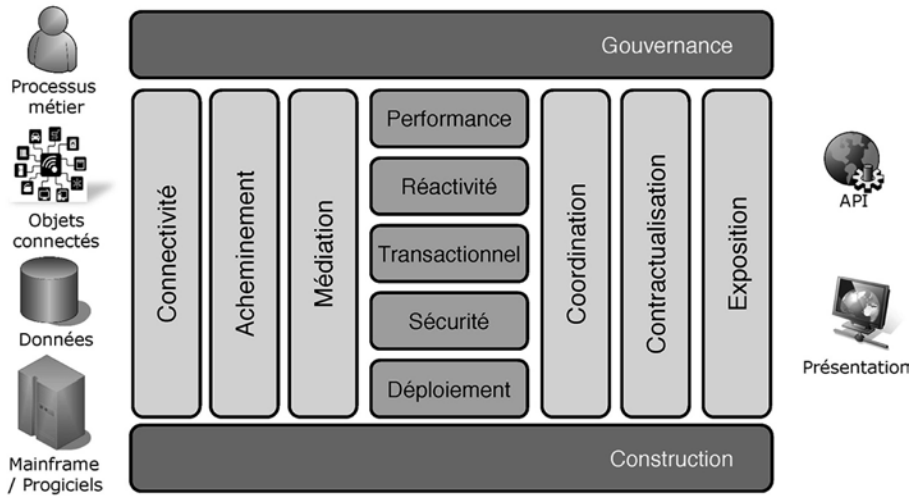
Cette capacité couvre l'édition et la gestion en configuration de code source et de descripteurs d'assemblages ou d'intégration de composants variés (connecteurs, tronçon d'intégration, interfaces, etc.), leur test et documentation, leur validation et leur paquetage au sein de différents conteneurs technologiques.

La gouvernance

Cette capacité couvre l'instrumentation des socles et des composants d'une architecture, le suivi des mesures, la gestion des alertes et le processus de correction ainsi que la conduite d'objectifs de valeur sur des transactions métiers ou d'objectifs de performance sur des ressources techniques.

La figure 2.2 résume les capacités techniques attendues par la DSI pour faciliter l'intégration de produits logiciels finis au sein du SI.

Figure 2.2 – La vue d'ensemble d'une chaîne des commodités techniques



12. Voir plus haut l'exigence métier sur l'expérience client.

2.2.3 Des exigences liées aux socles d'intégration

Dans la suite, les outils implémentant les capacités techniques verticales présentées en figure 2.2 seront désignés **socles d'intégration**.

◆ ***La connectivité doit devenir SMART***

Les applications sont trop longtemps restées passives face à leur nécessaire interconnectivité. Il s'agit désormais d'établir une proactivité à cette connectivité sans attendre ou dépendre d'un bus des communications à l'échelle de l'entreprise. Inéluctablement l'application qui gagne en renommée, doit se connecter à d'autres, qui à leur tour doivent se connecter encore à d'autres. Les contraintes d'interopérabilité entre de multiples technologies, ont fréquemment conduit au raccourci d'un connecteur Web. Cette banalisation du recours au canal de communication, tel que http notamment, ne doit pas faire oublier de reporter les contraintes non fonctionnelles de l'échange (voir plus bas les exigences sur les socles d'infrastructures) sur les connecteurs[13]. En particulier la nécessité prévaut de plus en plus d'une connectivité asynchrone à chaque application, qui permet de mieux préparer un eco-système communiquant à l'indisponibilité, la répétition en cas d'erreurs de communication, la temporisation en cas de pics de messages, l'ajout et la complémentarité de nouveaux participants sans impact sur ceux déjà communiquant etc.

◆ ***L'asynchronisme sous plusieurs formes***

La disponibilité des applicatifs interconnectés ne permet pas de supposer que chacun reste prêt à des sollicitations en permanence et en toutes circonstances. Il faut donc permettre l'acheminement de messages interapplicatifs sous plusieurs modes.

Zoom sur quelques modes d'échange

À la base des socles d'intégration, se place la capacité de véhiculer des messages entre des applications hétérogènes selon différents modes d'échange, différents formats, voire à différents niveaux d'intégration.

Les modes de transfert de messages à adresser sont généralement les suivants :

***Synchrone* (architecture couplée) :** le système appelant attend le transit aller/retour via le socle pour obtenir la réponse à sa requête avant de continuer ses traitements. *HTTP* est un protocole classique de l'échange synchrone.

A*synchrone* (architecture découplée) : la demande est placée dans un socle d'une file d'attente et le traitement en cours des participants à l'échange n'est pas interrompu. L'émetteur peut prendre en compte la réponse à sa requête dès sa disponibilité. Parmi les systèmes d'échange asynchrones les plus classiques, on peut citer les protocoles de *middlewares* orientés message (*MQSeries…*) ou les protocoles de messagerie d'entreprise (*SMTP…*).

13. Les anglo-saxons résument cela dans l'expression 'Smart end-points and dumb pipes'

***Publication / Souscription* (architecture décentralisée) :** dans ce contexte, le socle d'intégration met à disposition des abonnements à des thématiques de messages. Les systèmes s'abonnent aux changements via ces thématiques. Un cas classique est une souscription aux informations d'évolution d'un référentiel métier. Un exemple connu est le protocole *Tibco Rendezvous*.

Information par déclenchement : dans ce cas, c'est un événement applicatif qui provoque l'échange de messages. Le socle d'intégration détecte un changement au sein d'une des applications du SI et notifie les îlots applicatifs concernés. On peut citer l'exemple de l'arrivée d'un nouveau collaborateur dans l'entreprise : son insertion dans le logiciel de gestion des ressources humaines déclenche la création d'une boîte mail (provisioning SPML SCIM).

Dans le cas d'une architecture plus fortement distribuée, il peut être nécessaire d'envisager une intermédiation via un ou plusieurs socles d'intégration, voire de faire jouer à chaque application le rôle de nœud de médiation. On parle alors de communication de pair à pair.

Échange *Pair à Pair* (architecture distribuée) : dans ce cas, le socle d'intégration distribue ou fait distribuer à chaque système des tables d'encodages permettant d'identifier séparément les messages et les applications avec qui elle peut communiquer directement. De proche en proche, un graphe de communication s'établit pour permettre d'acheminer n'importe quel message à n'importe quelle application. Des exemples connus de ces protocoles sont par exemple Skype ou BitTorrent.

◆ *Reposer sur des référentiels de métadonnées et un tiers de médiation*

Pour offrir à la DSI une vision des ressources existantes et de leurs possibilités d'évolution et d'intégration, il est important de disposer d'un référentiel des différents applicatifs et dispositifs physiques. Ce référentiel doit décrire les dépendances entre ces différents applicatifs, ainsi que leurs points d'entrée et de sortie et pointer sur les dictionnaires des messages échangés entre applicatifs.

Pour assurer l'authentification et la gestion des droits des accédants, la plate-forme doit, en outre, disposer d'un référentiel des acteurs internes et externes au SI (personnes et applicatifs).

De plus, il peut être nécessaire de déployer un référentiel consolidé de description des données : il s'agit d'équivalence sémantique entre concepts métiers, de tables de références qui pointent vers les systèmes d'entreprise grâce à des clés d'identification uniques, etc. Ce référentiel permet en particulier de dupliquer et d'agréger des données métiers en conservant une maîtrise sur leur intégrité.

Enfin, au cours de processus mettant en jeu plusieurs applications, il peut être utile que la plate-forme joue un rôle de traducteur. La traduction à effectuer par le socle d'intégration consiste à transposer des messages d'une syntaxe donnée vers une autre en évitant d'y mélanger des règles métiers. Elle devra pour cela disposer des dictionnaires propres à chaque application, c'est-à-dire d'un référentiel des grammaires utilisées par chacune d'entre elles. Le besoin de traduction requiert donc un référentiel des dictionnaires de traduction.

◆ *Maîtriser la coordination des échanges et les couplages dans le SI*

L'intégration interapplicative requiert de choisir en connaissance de cause le degré du couplage à retenir entre les systèmes amenés à communiquer. Ces choix impactent directement les caractéristiques de l'architecture mise en œuvre et révèlent l'utilité de certains socles d'intégration. On distingue habituellement les exigences suivantes dans l'analyse du couplage plus ou moins fort entre plusieurs systèmes à intégrer :

- ✓ le mode de **formatage** (binaire, csv, XML, json…). Ce dernier impacte directement le niveau de compatibilité technologique requis ;
- ✓ le mode de **transfert** (synchrone, asynchrone, abonnement). Il situe la dépendance à une temporalité ;
- ✓ le mode d'**étiquetage** (document, appel distant, signal d'événement). Il influence plus particulièrement le type de programmation induit par l'échange ;
- ✓ le mode d'**acheminement** (une et une seule fois, au moins une fois…). Il joue sur la préservation de la cohérence, notamment en cas d'échec initial ;
- ✓ le mode de **diffusion** (broadcast, unicast, multicast). Ceci dirige la dépendance à une localisation notamment pour des volumes important ;
- ✓ le mode d'**expression** (énuméré, chorégraphié, orchestré, hyperlien[14]). Il permet de maîtriser un niveau d'harmonisation des échanges possibles à l'exécution ;
- ✓ le mode de **conversation** (requête/réponse, fire/forget, polling). Celui-ci dirige le degré de participation souhaité ;
- ✓ le mode d'**intermédiation** (proxy, coordinateur, pair à pair, relais). Ce dernier place l'échelle de la protection ou de l'étanchéité entre les participants à l'échange.

Chaque exigence de découplage requiert de consentir à un effort qu'il conviendra d'adapter à la situation rencontrée, aux moyens du projet mais aussi aux principes convenus à l'échelle d'un domaine d'activité, d'une organisation et du réseau concerné.

Par exemple, il peut se révéler précieux d'imposer plus d'exigences au découplage dans le franchissement de deux zones distinctes d'urbanisation du SI, plutôt qu'au sein d'un même quartier d'applications amenées à évoluer ensemble.

◆ *Faciliter la démultiplication des points d'accès*

Le SI ouvert doit désormais offrir des interfaces pour une diversité d'usages de plus en plus large. Il sert des typologies d'utilisateurs de plus en plus variées, comme des employés, des clients ou des fournisseurs, mais il doit aussi se préparer à dialoguer avec plusieurs typologies de machines, en particulier des logiciels externes à l'entreprise, des objets connectés, des applications dans le cloud et parfois aussi des communautés de développeurs eux aussi externes à l'entreprise souhaitant continuer de construire d'autres chaînes de valeurs à partir de certaines informations mises à disposition par le SI.

---

14. Voir les principes « HATEOAS » d'échanges basés sur la représentation d'états (REST).

La DSI doit donc offrir des mécanismes techniques permettant de mettre à disposition des interfaces Homme machine (IHM), ainsi que des interfaces programmatiques (API) destinées à faciliter des dialogues de machines à machines. Chaque population peut requérir une interface dédiée, mais pour autant, le SI ne peut réaliser autant de systèmes. Ce découplage franc entre interfaces ouvertes et systèmes réellement implémentés est à la base d'une construction SOA. Ces interfaces devenant les parties visibles du SI, doivent à leur tour satisfaire l'exigence métier de soin de « l'expérience client » (facilité d'accès à la documentation, à la souscription, etc.) pour pouvoir être facilement adoptées par les différentes populations clientes du SI selon leur environnement et leurs propres contraintes. Elles pourront aussi être accédées par des processus plus ou moins automatisés.

On verra dans cet ouvrage que le style SOA privilégie la notion de « *best effort* », c'est-à-dire que chaque application doit faire de son mieux pour présenter des interfaces haut niveau plus faciles à intégrer par les autres.

- ***Mesurer l'usage effectif et les ressources***

Ce contexte induit la nécessité de rendre également plus systématique l'instrumentation des logiciels et la consolidation de traces afin de disposer d'indicateurs pertinents sur le plan technique (pourcentage de saturation d'un espace de stockage, d'un CPU ou d'un réseau, nombre de redémarrages d'une prestation logicielle dans la semaine, nombre quotidien d'utilisateurs par prestation, etc.). Les équipes d'exploitation cherchent désormais à automatiser le plus possible les tâches de surveillance et de diagnostic des incidents, voire de résolution des problèmes les plus classiques. Elles doivent aussi pouvoir dimensionner et anticiper la montée en puissance des infrastructures à fournir (salle des machines, nombre de serveurs, équipes de piquet, etc.). Les socles d'intégration doivent prendre en compte ces exigences techniques de manière homogène et globale afin de minimiser le recours à des outils ou des compétences encore souvent trop spécifiques.

2.2.4 Des exigences liées aux socles d'infrastructure

Comme l'illustre la figure 2.2, d'autres capacités techniques horizontales ciblent des **socles d'infrastructure** qui complètent la chaîne des commodités techniques.

- ***Déployer automatiquement les composants applicatifs***

Déployer plus rapidement et industriellement dans l'infrastructure

À l'instar des acteurs du cloud, la DSI doit faciliter le recours à une mise en opération quasi **libre-service** et **automatisable** de la majorité de ses composants applicatifs. Ceci reste vrai *a fortiori* pour les socles technologiques des applications dont elle est fournisseur. Il n'est plus possible d'attendre plusieurs jours pour qu'une machine soit disponible et que

les composants logiciels qu'elle doit accueillir y soient installés, configurés et instanciés dans la bonne version sur les différents environnements souhaités. En outre, l'infrastructure doit préserver l'isolation de la sécurité et faciliter **une certaine extensibilité** face aux pics de charge, aux pannes du quotidien et selon la criticité ou le succès des composants métiers mis à disposition. Dans les cas les plus contraignants, cette extensibilité doit s'adapter automatiquement à l'intensité de la sollicitation. C'est déjà ce que propose des infrastructures comme celle d'Amazon Web Services.

La DSI avait traditionnellement recours aux technologies de virtualisation. Pour gagner en efficacité, elle doit désormais recourir à la mise en conteneur d'applications, mais aussi au cloud, notamment à l'IaaS, pour se prémunir d'une gestion à trop bas niveau des aspects multitenants, de distribution géographique ou d'adaptation à la demande des ressources CPU, mémoire et disque. Cette infrastructure peut s'établir de façon plus ou moins dédiée selon les contraintes légales de l'organisation ou du métier envisagé.

Ce besoin conduit aussi beaucoup de composants du marché, et en particulier les socles d'une « plate-forme SOA » (chapitre 6), à alléger leur empreinte technologique propriétaire pour laisser la possibilité de programmer leur installation, leur paramétrage ainsi que la configuration de leur déploiement.

Charge à la DSI de suivre ce rythme pour y recourir et faire de même avec les composants logiciels qu'elle implémente et souhaite déployer.

◆ *Contrôler les voies d'accès*

La mise à disposition de produits digitaux nécessite en amont des capacités techniques de gestion de la sécurité des échanges afin d'établir un cadre de confiance pour tous les clients de ces produits. On note ici que la digitalisation recherchée par les entreprises conduit désormais à considérer à la fois des clients humains, les utilisateurs, mais aussi de plus en plus des clients logiciels voire des machines ou des objets connectés à l'écosystème du SI. On retrouve ici le « mantra AAA » (Authentication – Autorisation – Audit) des protocoles de sécurisation d'échanges de messages.

Un socle d'infrastructure doit donc proposer une réponse aux problématiques suivantes :

- ✓ Authentifier les accédants au SI, en particulier si ceux-ci sont des personnes ou des machines externes à l'entreprise (clients, fournisseurs).
- ✓ Assurer la confidentialité et l'intégrité des échanges, en particulier si ceux-ci mettent en jeu des tiers externes à l'entreprise.
- ✓ Assurer la disponibilité des services.
- ✓ Assurer l'auditabilité des services : que s'est-il passé ?