Les pratiques SOA à éviter, Volume I

Comment empêcher votre architecture SOA de devenir un fardeau ?

Progress®
Actional®

Sommaire

Pour une mise en place réussie de votre architecture SOA

Combien de fois avez-vous entendu dire : "La sécurité de votre architecture SOA doit être suffisamment stricte, mais pas trop rigide afin de ne pas affecter la souplesse et les performances", et pensé :
"Oui, mais comment mesurer cela, et où mettre le curseur ?"

Vous n'êtes pas le seul, et nous ne disons pas cela pour vous rassurer. Ces pratiques SOA fournissent des conseils et une valeur ajoutée non négligeables, mais les appliquer à votre entreprise exige une certaine expérience. En effet, les bonnes pratiques s'avèrent parfois inadaptées à votre situation ou modèle économique spécifique.

Chez Progress Software, nous pensons qu'il est temps d'adopter une nouvelle approche pour concevoir votre système d'information.

Votre meilleure idée peut s'avérer être votre pire pratique SOA

Depuis plus de six ans, Actional de Progress est à l'avant-garde dans la gestion des services web ; et fournit la visibilité, la sécurité et le contrôle des activités de service et des processus. Nous avons donc eu le temps d'en considérer tous les aspects, et avons résolument mis au placard quelques-unes de nos propres fausses bonnes pratiques.

Nous nous sommes également rendu compte que savoir ce qu'il ne fallait pas faire lors de la conception d'une architecture SOA, pouvait s'avérer extrêmement utile.
C'est pourquoi nous avons créé *Les pratiques SOA à éviter*, Volume I.

Vous découvrirez dans ce guide des erreurs réellement commises par vos pairs.[1]
Nous vous expliquerons ce qui se cache derrière ces concepts, les raisons pour lesquelles ils ont échoué, et les approches vous permettant d'obtenir de meilleurs résultats.
(En d'autres termes, les principes de l'architecture SOA n'auront véritablement plus aucun secret pour vous.)

Peut-être envisagez-vous de mettre en place, globalement ou partiellement, une architecture SOA ?
Ou êtes-vous déjà bien avancé dans ce processus ? Quoiqu'il en soit, nous vous conseillons de garder ce guide à portée de main.
Il y a très probablement une mauvaise pratique mentionnée ici, que vous ou l'un de vos collaborateurs étiez sur le point d'adopter.

[1] Bien que nous ayons modifié les noms de toutes les entreprises citées dans ce document (à l'exception du nôtre), les cas présentés sont réels et difficiles.

Les 8 pratiques SOA à éviter

Les mauvaises pratiques SOA sont évidemment très nombreuses, mais nous en avons sélectionné 8 qu'il convient à tout prix d'éviter. Mais ne vous y trompez pas, il ne s'agit là que du sommet de l'iceberg…

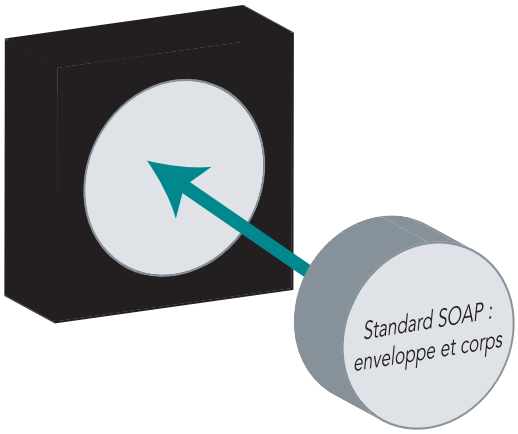
8. Je sais que SOAP est un standard. Pourquoi ne pourrions-nous y apporter quelques modifications ?

La performance est l'une des principales préoccupations des environnements informatiques, et notamment des services Web. Les messages de service Web transitent par plusieurs intermédiaires, et chaque message doit être traité dans son intégralité (analyse de l'ensemble du message), ce qui peut affecter la vitesse de transmission. Mais, comme le démontre cette mauvaise pratique, altérer un standard ne fera qu'aggraver vos problèmes.

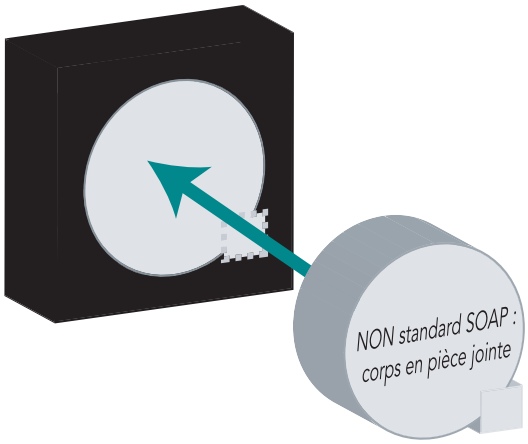
L'idée de départ

Désireuse d'accélérer la vitesse de transmission des messages de différents services Web, l'équipe informatique de Hutchings Healthcare a décidé de diviser les messages SOAP en deux parties : l'enveloppe d'une part, le corps de l'autre. L'enveloppe était transférée de manière habituelle, mais le corps était toujours transféré en tant que pièce jointe, au lieu de respecter le standard SOAP et d'être transféré dans le message.

Standard SOAP



Standard SOAP utilisé de manière non orthodoxe



Pourquoi c'était une erreur

L'équipe informatique utilisait le standard SOAP, mais d'une manière non orthodoxe. En conséquence, les messages et applications étaient incompatibles avec les autres mises en œuvre basées sur des standards. Même si la société prétendait qu'il s'agissait d'une architecture SOA standard, elle avait en fait créé une solution de messagerie propriétaire.

Elle a supposé à tort qu'elle pouvait disposer d'une architecture SOA tout en utilisant des "extensions" propriétaires obligatoires. Voici pourquoi cela ne peut pas fonctionner :

1. Ces extensions affectent l'interopérabilité, ce qui limite considérablement les performances de votre architecture SOA et augmente significativement les coûts.
2. Cette solution de messagerie étant propriétaire, du code spécifique doit résider à chaque extrémité de la connexion. Vous devez donc personnaliser l'ensemble de vos applications et logiciels serveur.

3. Ces extensions vous empêchent d'utiliser efficacement vos outils de développement, qui sont optimisés autour des standards SOAP.

4. La gestion des versions est plus difficile. Lorsqu'un éditeur mettra à jour ses logiciels, la solution de messagerie devra probablement être adaptée à la plate-forme révisée.

SOAP étant un langage ouvert, Hutchings Healthcare a altéré sa valeur intrinsèque en le modifiant. Si SOAP ne fonctionnait pas dans son architecture SOA, la société aurait mieux fait de participer à la mise à jour du standard plutôt que de créer une solution propriétaire.

Une meilleure approche

La performance tient à la manière dont vous allez concevoir votre architecture SOA. Une architecture SOA digne de ce nom vous permet d'utiliser des standards ouverts, facilitant la réutilisation et la souplesse. Vous n'êtes donc pas dépendant d'un fournisseur ou d'une technologie spécifique. Même les systèmes les plus hétérogènes communiquent rapidement et facilement sur diverses plates-formes logicielles et matérielles. Cependant, tous ces avantages sont réduits à néant lorsque vous introduisez une solution propriétaire, comme ce fut le cas pour Hutchings Healthcare lorsqu'il a modifié le standard SOAP.

Lors de la conception de votre architecture SOA, évitez "le jeu du téléphone", c'est-à-dire un nombre trop important de composants et services transférant le message et alourdissant la charge du système. Au lieu de cela, assurez-vous que les composants sont faiblement couplés et utilisent un langage universel. De cette manière, ils pourront sauter les étapes A à D, et passer directement à l'étape E pour obtenir les données demandées ou initier le processus correct. Il vous suffit ensuite de traiter les données dont vous avez besoin.

7. Une architecture SOA immédiate : prenez vos applications existantes et encapsulez-les !

Une architecture SOA ne doit pas être complexe ; toutefois, il est primordial de disposer d'une stratégie SOA claire et de maîtriser les risques de sécurité et les moyens de les limiter. Dans le cas contraire, vous pourriez courir tout droit vers la catastrophe.

L'idée de départ

Granger Machinery, fabricant d'équipements industriels, pensait avoir eu une bonne idée en mettant en œuvre une architecture SOA. La société prévoyait d'utiliser des services Web, afin de permettre à d'autres lignes d'activité d'accéder à ses bases de clients et de stocks. L'architecture SOA a été initiée en permettant à un client d'envoyer des instructions SQL complètes dans une requête adressée à la base de données. Cependant, ce protocole rendait la base de données relationnelle vulnérable aux attaques par injection SQL, dans lesquelles les pirates contournent les instructions SQL définies sur un serveur Web afin d'injecter les leurs.

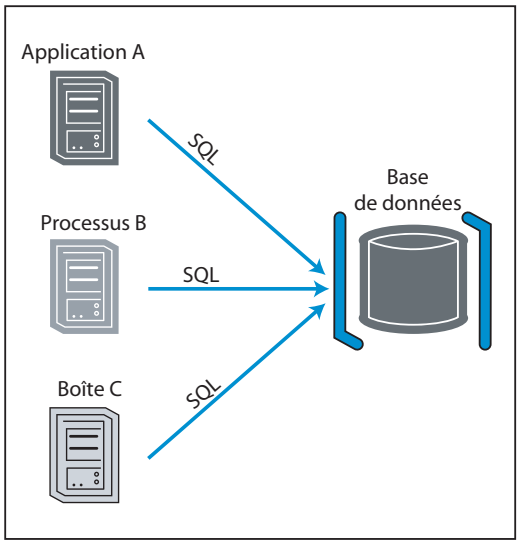
L'équipe informatique pensait pouvoir réduire ce risque en dotant son système de sécurité existant d'une fonctionnalité anti "attaque par injection". Cette fonction devait détecter et supprimer les instructions SQL écrites pour exécuter des opérations destructives du type "Drop" ou "Delete".

Pourquoi ce n'était pas judicieux

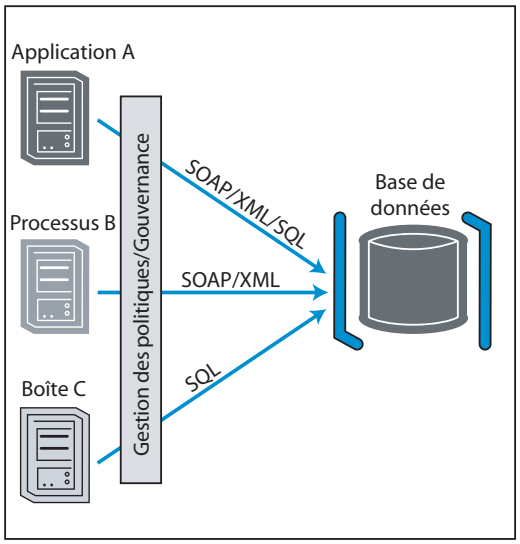
Cette approche de service Web, consistant à placer un agent dans une base de données ou une application existante, n'est pas dangereuse en soi. Toutefois, si vous y regardez de plus près, cette forte imbrication implique que le demandeur connaisse les détails de mise en œuvre pour pouvoir passer l'appel SQL. Elle présente deux problèmes majeurs :

1. Elle n'inclut pas de contexte métier. L'équipe informatique doit associer un contexte métier (ou règles) au niveau des services Web, au lieu d'autoriser des requêtes SQL "passives". Ces règles définissent les personnes autorisées à placer une requête dans la base de données, et les cas précis dans lesquels elles peuvent accéder aux données. Les individus doivent donc s'authentifier avant d'accéder à la base de données. Celle-ci ne donne plus d'informations à l'aveuglette, et les requêtes destructives ne peuvent plus accéder directement à la table.

2. Les problèmes d'injection SQL sont dus à deux causes différentes : données néfastes injectées dans des instructions SQL, et instructions SQL néfastes elles-mêmes. Les fonctionnalités de protection contre les attaques par injection précédemment décrites, ne permettent pas de résoudre ces problèmes.



En cas de communication directe avec la base de données, les applications et processus ont trop de pouvoir sur les données.



Le déchargement de la gestion des politiques et de la gouvernance permet de faciliter la gestion et la mise à jour des règles et des politiques.

Une meilleure approche

L'application d'une architecture SOA et de standards de service Web vous permet d'atteindre rapidement les objectifs de réutilisation et d'intégration. Cependant, cette approche doit faire partie d'une stratégie SOA globale, faute de quoi vous risquez de rencontrer des problèmes au niveau application.

Dans le cas de Granger Machinery, la société devait répondre aux questions suivantes dans sa stratégie SOA :

- Quelles personnes doivent accéder aux données ?
- Comment doivent-elles y accéder ?
- Pouvons-nous et devons-nous prioritiser les requêtes et transactions ?

Exemple : Problèmes au niveau application

Si différents services sont autorisés à communiquer entre eux, un individu non autorisé peut extraire des informations ou utiliser des fonctionnalités auxquelles vous ne souhaitiez pas qu'il accède. Imaginons que Granger Machinery ait disposé d'un portail de recrutement, et que d'autres lignes d'activité aient souhaité utiliser ces informations, même si elles n'étaient pas autorisées à le faire. Ce service en accès libre aurait pu faire courir à la société un risque de violation de la sécurité et des lois sur la confidentialité des données, et ralentir les performances.

Le résultat ? L'intégration d'un service est un excellent moyen de réutiliser les applications qui vous fournissent déjà de la valeur. C'est par conséquent une méthode excellente pour concevoir une architecture SOA, du moment que vous disposez d'une stratégie permettant de déterminer les services à intégrer, et les règles qui régissent leur utilisation.[2] Toutefois, intégrer les services de manière aléatoire peut entraîner des problèmes de sécurité et de performance, à l'intérieur et à l'extérieur de l'entreprise.

Souvenez-vous également que la conception d'une architecture SOA est un processus progressif. Vous pouvez commencer par intégrer des services qui fonctionnent correctement aujourd'hui, et en reconcevoir d'autres par la suite. Par ailleurs, vous pouvez prioritiser les services à reconcevoir et ceux à créer entièrement, en fonction des nouvelles opportunités qui vous sont offertes.

[2] La gouvernance doit s'assurer que ces questions sont prises en compte et intégrées dans la stratégie. Les politiques assureront ensuite le contrôle, la direction et la gouvernance d'exécution garantissant leur application et leur mesure.

6. Les pirates ne peuvent rien contre nous, nous avons un pare-feu XML !

L'interopérabilité est un facteur déterminant pour communiquer avec les systèmes de vos partenaires et clients. Il est par conséquent très important de garder en tête que la conception d'une architecture SOA peut avoir un impact significatif sur ces groupes. Comme le démontre cette mauvaise pratique, les meilleures intentions peuvent parfois engendrer les pires effets, et l'ignorance n'est pas une bonne chose dans le monde SOA.

L'idée de départ

Dane Foods s'enorgueillissait d'offrir à ses clients un service de pointe, en plus de la fabrication de produits cuisinés et congelés de qualité. Son département Service client était une véritable boutique Oracle, des bases de données back-end, aux systèmes Web et Oracle frontaux. Ce système n'était pas parfait mais fonctionnait parfaitement depuis plusieurs années.
Le département Service client était autonome, à l'exception d'un nombre limité d'interactions avec les clients et partenaires externes pour la collecte des réclamations. Les réclamations étaient acceptées dans quasiment tous les formats, car le département ne souhaitait pas en imposer un qui l'empêcherait d'obtenir des informations précieuses. Aucun autre processus de fonctionnement ne dépendait des interactions avec les partenaires externes.

Le PDG de Dane Foods a décidé que la société avait besoin d'une architecture SOA. Il n'en savait pas grand-chose, mais il en avait entendu parler. Et cela lui semblait être une bonne idée.
Puis, l'équipe informatique a souhaité acheter un pare-feu XML pour protéger deux services qui effectueraient la validation de schéma sur plusieurs dizaines de transactions par seconde. Le Directeur informatique a alors lancé un appel d'offre, qui plaçait l'amélioration des performances et de l'interface utilisateur au premier plan des objectifs de l'architecture SOA.

Une fois le pare-feu XML installé, l'équipe informatique a été confrontée à des problèmes de communication qui ont entraîné la frustration des consommateurs et partenaires. Il a alors été obligé de désactiver plusieurs fonctions du pare-feu pendant qu'il essayait de résoudre les problèmes.

Pourquoi ce n'était pas judicieux

L'idée de Dane Foods a échoué pour trois raisons :

1. L'appel d'offre confondait les exigences de performances et d'interface utilisateur du pare-feu, avec les objectifs de l'architecture SOA. Non seulement ces exigences ne font pas partie des objectifs SOA, mais elles ne sont pas non plus considérées comme des atouts du pare-feu.
2. Le département devait sécuriser l'interopérabilité avec ses utilisateurs externes.
3. L'équipe informatique autorisait des schémas de données différents provenant de sources diverses, ce qui lui a causé des problèmes et a par la suite obligé les partenaires et les clients à modifier leur schéma. Cela a généré du travail et des coûts imprévus pour ces entreprises.

Une meilleure approche

Premièrement, assurez-vous de développer une architecture SOA pour les bonnes raisons. Celles-ci doivent être basées sur votre activité et vos objectifs informatiques.

Deuxièmement, concevez une stratégie adaptée, ainsi qu'un moyen de communiquer vos plans SOA aux partenaires, clients et acteurs concernés.

Troisièmement, sachez que l'interopérabilité est une exigence de l'architecture SOA, mais qu'elle n'apparaît pas par magie. Les services Web incluent souvent de nombreux fournisseurs et consommateurs. Avant de corriger ou de mettre un service à jour, vous devez connaître l'ensemble des interdépendances (consommateurs en aval et fournisseurs en amont) et l'impact éventuel sur l'activité.[3]

[3] Les solutions Actional vous permettent de les détecter et de les découvrir automatiquement.

5. Le scénario SOA extrême : on efface tout et on recommence !

La consolidation des systèmes peut être une bonne méthode pour permettre aux entreprises d'améliorer l'efficacité et de réduire les coûts, tout particulièrement en cas d'acquisition. Mais, comme le démontre cette mauvaise pratique, la migration et l'intégration des systèmes dans une architecture SOA ne doit pas se faire trop vite et en une seule fois. Et vous ne devez pas choisir un cabinet de conseil sans avoir auparavant analysé ses compétences.

L'idée de départ

Carson Office Supplies se développait de manière exponentielle avec l'acquisition en moins d'un an de 20 franchises auparavant indépendantes. La société a bientôt été confrontée à un chevauchement important de ses systèmes et processus. Elle a donc décidé d'intégrer certains d'entre eux en concevant une architecture SOA. Carson Office Supplies avait surnommé cette initiative le "Project TROY".

La société a fait appel au cabinet Wecan Consulting, qui a conçu un programme complet impliquant une équipe de 160 à 200 personnes, et des logiciels développés par 30 fournisseurs différents, dont :

- Actional Looking Glass and SOAPstation de Progress Software
- Adobe Form Server (maintenant appelé Adobe LifeCycle Forms)
- BEA WebLogic
- IBM Rational Suite
- ILOG JRules
- Informatica PowerCenter and SuperGlue
- IONA Artix
- Microsoft SharePoint Server
- Network Inference Cerebra Server
- Oracle Application Server
- Oracle eBusiness Suite
- Plumtree Corporate Portal
- RSA ClearTrust
- Software AG Tamino XML Server
- Sunflower Assets

Wecan Consulting était constamment en retard sur le programme et ne respectait pas les étapes initialement prévues. Après une réduction massive de l'équipe, des tentatives de replanification et la révision des exigences, Wecan Consulting a été obligée de recommencer et de reconcevoir entièrement l'architecture SOA.

Le nouveau programme de Wecan Consulting consistait à adopter une stratégie à base de progiciels avec des intégrations point-à-point. Mais le moral n'était pas au beau fixe au sein de l'équipe informatique de Carson Office Supplies. Après ce faux départ, l'équipe portait peu d'intérêt à la poursuite du développement d'une architecture SOA, et montrait peu d'enthousiasme. Carson Office Supplies a dû patienter et a connu bien d'autres soucis avant de pouvoir mesurer les avantages d'une architecture SOA.

Pourquoi ce n'était pas judicieux

La technologie SOA vous permet de bénéficier d'une approche de pointe et de tirer le meilleur parti des standards ouverts. L'utilisation de technologies multifournisseurs, utilisant des standards ouverts, ne devrait en principe poser aucun problème. Cependant, le Project TROY essayait d'en faire trop, et trop vite.

Même si un grand nombre de choses ont été faites correctement, la complexité du Project TROY était écrasante, du champ d'application au nombre de fournisseurs impliqués. Carson Office Supplies avait besoin d'une approche plus simple et plus progressive.

Pour compliquer les choses, Wecan Consulting, bien qu'étant un cabinet remarquable, n'avait pas beaucoup d'expérience dans les technologies SOA, et apprenait au fur et à mesure de l'avancement du projet.

Enfin, Carson Office Supplies a été amenée à renoncer à une véritable architecture SOA pour un modèle point-à-point, après s'être entendu dire que celui-ci lui apporterait les mêmes avantages. Au final, la société s'est retrouvée avec une solution d'intégration de niveau inférieur et à fort couplage, ce qui ne caractérise en rien une architecture SOA. Même si cette stratégie de remplacement a permis à l'ensemble des systèmes de "communiquer", elle n'a pas fourni le même niveau (et en conséquence, la même valeur) de réutilisation et de souplesse qu'une véritable architecture SOA.

Une meilleure approche

Fort heureusement, la technologie SOA vous permet d'éviter une approche de type "big bang" ou "remplacement global". Si vous envisagez de migrer vers une architecture SOA, posez-vous les questions suivantes :

- Qu'est-ce qui fonctionne ?
- Qu'est-ce qui ne fonctionne pas ?
- Qu'est-ce qui fournit de la valeur ?
- Quelle nouvelle opportunité devons-nous poursuivre ?

Votre stratégie doit couvrir la migration de l'ensemble de votre système vers l'architecture SOA, mais gardez à l'esprit que ce processus peut prendre plusieurs années. Il est préférable de commencer avec des applications qui ne fonctionnent pas. Leur migration ou reconception offrira alors des avantages immédiats.

Par ailleurs, évitez de remplacer des applications qui fonctionnent correctement et fournissent de la valeur ajoutée. Vous pouvez immédiatement les intégrer dans votre architecture SOA, en plaçant simplement un adaptateur au niveau de celles-ci.

Puis laissez vos besoins déterminer quand et comment concevoir de nouvelles applications ou fonctions. Essayez d'éviter tout effort ne vous conférant pas de valeur immédiate, tel que la reconception ou le redéveloppement d'une application qui fonctionne déjà correctement. Au lieu de cela, assurez-vous que votre stratégie SOA tire le meilleur parti des processus qui vous confèrent le plus d'avantages. Puis analysez les applications et composants sous-jacents de chacun de ces processus. Si certaines applications fonctionnent correctement, placez simplement un agent au niveau de celles-ci. Si elles fonctionnent mal, ou si vous souhaitez améliorer le processus, créez un nouveau service permettant de fournir cette fonctionnalité.

Enfin, lors de la sélection d'un cabinet de conseil, procédez à l'analyse complète de ses compétences. La taille et la réputation d'un cabinet ne signifient pas nécessairement qu'il dispose de l'expérience nécessaire dans la technologie SOA pour pouvoir répondre à vos besoins.

4. Laissez tomber le clonage humain, clonez vos applications !

Le clonage et la reconfiguration des applications existantes peuvent sembler être une bonne méthode pour réaliser des économies sur les coûts de maintenance et de développement. Mais en pratique, c'est loin d'être le cas.

L'idée de départ

L'équipe informatique de Jesper Travel, un service de voyages en ligne, avait une approche novatrice en matière de gestion des systèmes. Elle a créé de nouvelles applications et de nouveaux processus en reconfigurant et en clonant des applications existantes. L'une d'elles, par exemple, avait 30 instances "distinctes" en production, toutes étant personnalisées à un certain niveau, avec 50% à 60% des fonctions de base dans chacune d'elles.

L'équipe informatique pensait que cette approche permettrait de réduire les délais de développement, de simplifier la maintenance et d'abaisser les délais de résolution des problèmes. Leur raisonnement était le suivant :

- Davantage de développeurs seraient familiarisés avec un plus grand nombre d'applications.
- L'architecture était simple, les deux points étant l'application clonée et le client.
- Il serait aisé de planifier et de calculer les coûts futurs, car un nouveau client = nouveau matériel + application clonée + quelques modifications spécifiques à ce client.
- Un nouveau client serait un modèle économique autonome, avec tous les coûts associés à celui-ci et pas d'infrastructure partagée à amortir.

Pourquoi ce n'était pas judicieux

Cette approche présentait quatre principales faiblesses. Premièrement, en cas de bogue dans le code de base, il était répliqué. Deuxièmement, le code de base était modifié lors du processus de personnalisation, mais ces modifications étaient rarement documentées. Donc, lorsque l'équipe de maintenance avait besoin de déterminer la cause d'une panne, elle y passait beaucoup plus de temps qu'à résoudre le problème lui-même.

Troisièmement, ce clonage générait une infrastructure, une maintenance et un développement redondants, tant au niveau commercial qu'au niveau informatique. L'équipe informatique n'avait aucun moyen de partager les ressources entre les clients, d'abaisser le TCO, ou de gérer les accords de niveau de service. (Mais les éditeurs de logiciels et les fournisseurs de matériel étaient assez satisfaits de ce modèle, car chaque nouveau client de Jesper Travel effectuait des achats de matériels et logiciels supplémentaires.)

Exemple : Intégration et simplification

Au lieu d'extraire un ensemble de règles de sécurité s'appliquant à l'ensemble des clients, tel que "toutes les données clients doivent être cryptées", Jesper Travel intégrait cette règle dans chaque application. Cela compliquait considérablement la gestion des systèmes. La simplification aurait permis à Jesper Travel de changer les règles en modifiant simplement la configuration de la fonctionnalité de base, au lieu du code de chaque application.

Quatrièmement, les modifications du code de base affectaient chaque client s'appuyant sur ces services. A la suite de chaque modification, les modules de base et les modules personnalisés associés devaient être testés avant que la société puisse redéployer ces services. En outre, l'équipe informatique ne pouvait pas se contenter de corriger les applications, mais devait passer par de nombreuses étapes, ce qui se traduisait par un cycle de mise à jour plus long. D'autre part, cette approche intégrait des règles, ou politiques, en tant que code dans l'application, au lieu de les "extraire". Cela ne faisait qu'accentuer la redondance et rendre la modification encore plus difficile.

Une meilleure approche

Si vous souhaitez économiser sur les coûts de maintenance et de développement, optez pour la réutilisation. Contrairement au clonage et à la reconfiguration, la réutilisation permet de développer votre activité tout en minimisant la complexité de vos systèmes informatiques.

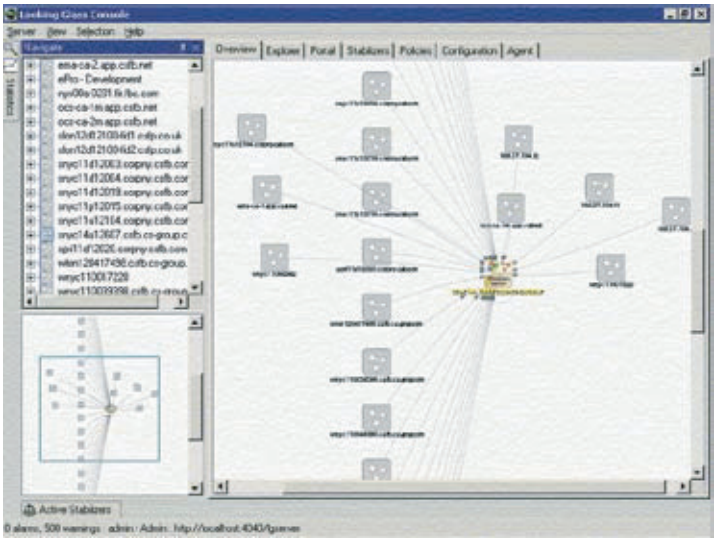
En concevant une architecture SOA, vous réduisez la redondance tant au niveau des applications que de l'infrastructure. Par exemple, vous pouvez extraire un ensemble limité de fonctions de base à partir de plusieurs centaines d'applications en double. Ces services réutilisables peuvent être utilisés globalement, sur l'ensemble des applications et processus, et permettent ainsi d'améliorer l'efficacité d'exploitation et de réduire les coûts de maintenance. Par ailleurs, vous pouvez standardiser, décomposer (extraire) et contrôler la fonctionnalité d'infrastructure de base des processus stratégiques, dont la sécurité, l'identité, le profil, le courtage et la mise en file d'attente des messages, la conversion, le service d'annuaire, etc.

Une architecture SOA vous permet également de concevoir de nouvelles solutions d'entreprise plus rapidement qu'en clonant des applications. Vous pouvez donc consacrer davantage de temps à personnaliser les solutions de chaque client, tout en tirant le meilleur parti de votre jeu de services de base. Ces services de petite taille vous aident également à réduire vos coûts logiciels et matériels, car ils permettent à votre entreprise d'optimiser ses actifs existants.
En outre, vous pouvez mesurer, gérer et sécuriser ces services, ou les modifier facilement à mesure que les politiques évoluent.

3. Même un enfant peut utiliser notre WSDL

La capacité de réutilisation est l'un des atouts clés de la technologie SOA, mais le contexte doit déterminer comment ces services peuvent et doivent être réutilisés. Comme le démontre cette mauvaise pratique, des problèmes de sécurité, de confidentialité des données et de conformité peuvent apparaître lorsque l'utilisation d'un WSDL (Web Services Description Language) n'est pas correctement régie.

L'idée de départ



Venton Automotive, un OEM spécialisé dans la fabrication de toits ouvrants, avait doté son département RH d'un nouvel outil remarquable. La société avait conçu son portail RH interne de manière à permettre aux employés de saisir et gérer eux-mêmes leurs propres informations, telles que qualifications, numéros de téléphone et sites, noms des responsables, etc.
De même, le département RH validait ces informations et supprimait manuellement les enregistrements des individus ne faisant plus partie de la société.

Ce portail fonctionnait en grande partie sur un système de confiance, avec une simple validation mineure des données, telles que structure hiérarchique, fonctions, etc.

Ce portail est bientôt devenu le seul endroit où l'on pouvait trouver des informations actualisées et complètes sur les employés, et où le département RH pouvait identifier un organigramme à jour. Au fur et à mesure, d'autres départements ont découvert ce portail et ont commencé à l'utiliser. L'équipe d'audit, par exemple, utilisait ses informations pour s'assurer que chaque employé avait été formé aux aspects de conformité et de confidentialité des données. Et l'opérateur téléphonique de l'entreprise s'en servait pour vérifier que les appels étaient acheminés vers les destinataires appropriés.

Lorsque d'autres développeurs souhaitaient accéder aux informations du portail RH, l'équipe informatique partageait le WSDL utilisé pour définir les formats et protocoles du service. Elle faisait sans réticence, car elle partait du principe que l'atout d'une architecture SOA était de disposer d'un service réutilisable. Le développeur principal partageait le WSDL avec trois ou quatre collaborateurs, qui à leur tour le partageaient avec d'autres développeurs. Au bout de quelques mois, nul n'était capable d'indiquer de manière exacte le nombre d'utilisateurs de ce service.

Pourquoi ce n'était pas judicieux

L'équipe informatique a bientôt appris que quelqu'un avait placé le WSDL dans une bibliothèque, qu'il avait ensuite partagée avec l'ensemble des équipes de développement. Elle a ensuite découvert que le service Web comptait plus de 30 utilisateurs.[4]

L'équipe informatique ne savait pas du tout qu'autant d'utilisateurs et lignes d'activité était pris en charge. Elle ne savait pas non plus :

- Si les départements appropriés étaient pris en charge ?
- Si les données étaient protégées ou cryptées ?
- Quelle était la capacité du service ?
- En cas de défaillance, quelles seraient les incidences ?
- Ce qui se passerait si elle avait besoin de mettre à niveau le service ?

Pour aggraver les choses, les développeurs utilisaient la bibliothèque en production, et l'équipe informatique avait donc un serveur de développement supportant des applications de production. Cela générait des risques sérieux en termes de sécurité, et mettait en péril les lois de conformité et de confidentialité des données.

Pour résumer, la direction informatique exposait ses systèmes à des risques importants, tout en manquant l'opportunité de montrer la valeur d'une architecture SOA, comme cela est démontré par la réutilisation.

Une meilleure approche

Pour obtenir une réutilisation sécurisée des services via une architecture SOA, vous devez disposer d'un processus détaillé. Celui-ci doit :

- Identifier les consommateurs et les fournisseurs qui s'appuient sur le service
- S'assurer de la mise en place de mesures de sécurité appropriées, de l'application de la politique informatique et des règles métier
- Mettre en place la technologie appropriée en matière de gestion et de gouvernance d'exécution
- Valider que le service s'exécute comme prévu et répond aux objectifs pour lesquels il a été conçu
- Inclure un mécanisme d'avertissement et une fonctionnalité de détection vous permettant d'être immédiatement informé de tout dérapage des opérations de production ou autre, tel que le partage inapproprié d'un service ou d'un WSDL

Le résultat ? Partager un WSDL peut sembler anodin, mais si cela n'est pas contrôlé, un grand nombre d'individus peuvent accéder à un service précieux et exposer ainsi vos données aux risques.

[4] La technologie Actional, qui détecte automatiquement tous les services utilisés, a permis de révéler le nombre d'utilisateurs de ce service Web.

2. Les pirates ne peuvent atteindre nos données, nous effectuons une validation de schéma externe !

La sécurisation de vos systèmes contre les pirates est une bataille permanente. Effectuer une validation de schéma externe peut sembler être une simple mesure de protection supplémentaire, toutefois cette approche accroît réellement votre vulnérabilité aux attaques.

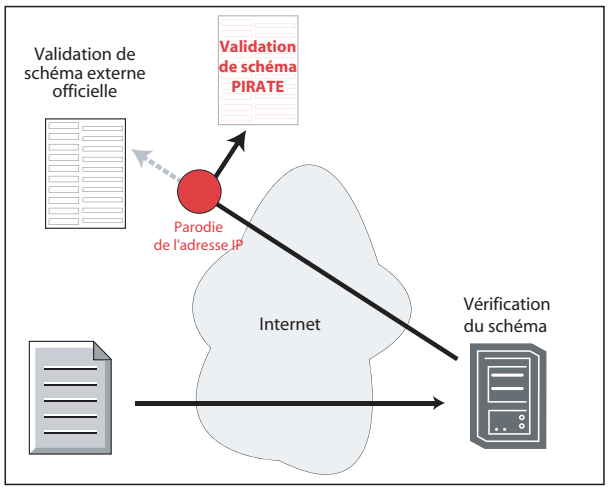
L'idée de départ

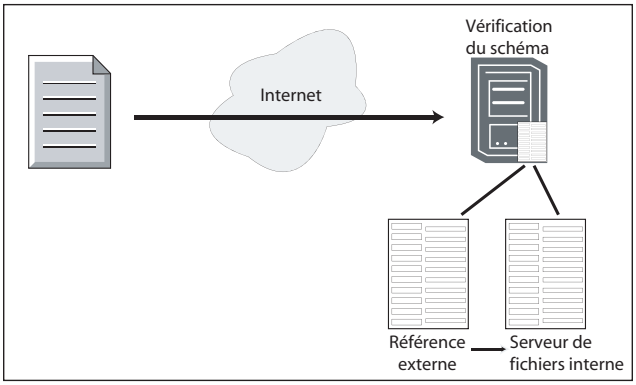
ABS (Academic Business Services), un groupe d'établissements spécialisés dans la formation universitaire des salariés, souhaitait améliorer la sécurité de ses services Web. Il a décidé d'effectuer une validation de schéma externe. Cela permettait à la "vérification" de schéma de résider dans un autre emplacement et de réduire ainsi le risque que les pirates puissent déchiffrer le schéma à l'aide de transactions "de sondage" soigneusement conçues.

Pourquoi ce n'était pas judicieux

Bien souvent, l'emplacement de la validation externe peut être déterminé à partir de la réponse d'une transaction. Un pirate a rapidement découvert qu'il pouvait parodier l'adresse IP d'ABS et modifier l'emplacement de la vérification de schéma de la société. Il pouvait alors lancer des transactions avec un autre schéma, et le faire vérifier par la validation de schéma pirate résidant à l'adresse IP parodiée. Cela provoquait l'échec des transactions légitimes.

Bien que les pirates n'aient pas réussi à obtenir d'informations importantes d'ABS, ils sont quand même parvenus à affecter son activité.





Une meilleure approche

Vous pouvez assurer la protection de votre schéma en suivant quelques étapes simples. Tout d'abord, communiquez-le à vos partenaires, afin de pouvoir les contacter directement. Puis, placez la validation de schéma dans une table interne sécurisée par des mesures de défense de périmètre traditionnelles et une sécurité d'extrémité.

1. Notre initiative SOA est un franc succès : regardez le nombre de services que nous proposons !

A l'ère de la rentabilité, la plupart des sociétés souhaitent pouvoir s'appuyer sur des chiffres pour démontrer la progression et la valeur conférée par leur initiative SOA. Comment quantifiez-vous réellement la valeur de la technologie SOA ? Un conseil : ne vous contentez pas de calculer le nombre de services Web.

L'idée de départ

Le conseil d'administration de Rimstar Bank avait demandé un rapport sur la valeur de son initiative SOA, mais le directeur technique de la banque ne savait pas comment la quantifier. Finalement, il a décidé de baser son appréciation sur le nombre total de services que l'équipe informatique avait développés et qui étaient en production. La société disposait de 25 services en production, 5 d'entre eux étant réutilisés pour plusieurs processus.

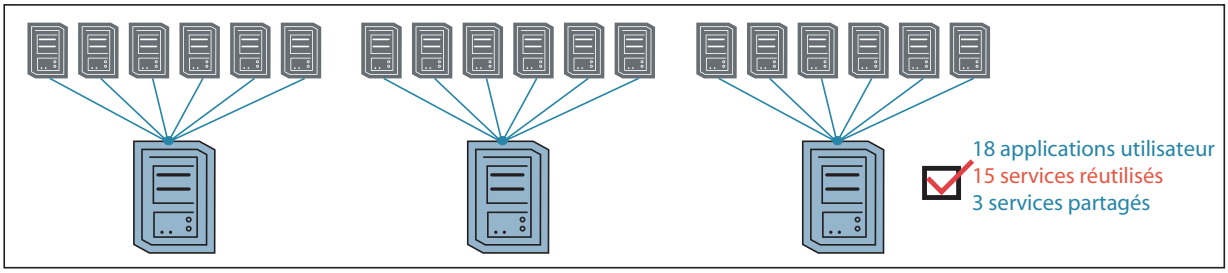
Pourquoi ce n'était pas judicieux

La principale valeur fournie par une initiative SOA est la réutilisation. Le nombre de services Web n'a donc rien à voir dans son succès. En effet, un nombre élevé de services peut être un signe d'échec, car il est fort probable qu'il y ait alors peu de réutilisation.

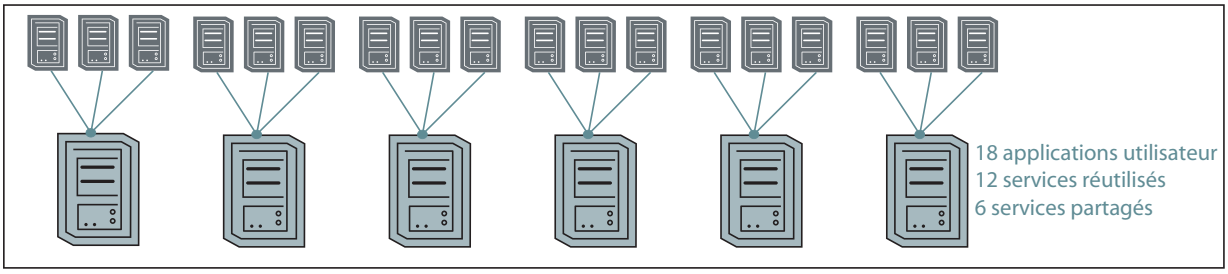
Chez Rimstar Bank, la réutilisation était faible, 20 des 25 services en production étant toujours utilisés en tant qu'applications individuelles. Ainsi, les applications existantes converties en services, ou les nouveaux services créés, ne pouvaient pas être pris en compte dans un rapport de rentabilité. En effet, ils n'étaient pas réutilisés par d'autres applications ou processus, ni par d'autres lignes d'activité.

Quantification de la valeur de la technologie SOA – Quelle est l'initiative la plus performante ?

#1



#2



Une meilleure approche

Pour estimer la valeur d'une architecture SOA, il vaut mieux se concentrer sur le nombre de services Web existants réutilisés que sur le nombre de services Web récemment créés. La réutilisation n'est pas seulement un avantage stratégique de l'architecture SOA, c'est également un aspect que vous pouvez quantifier.

Vous pouvez mesurer la fréquence d'utilisation d'un service et le nombre de processus qu'il supporte, et ainsi le nombre d'éléments réutilisés. Cela vous permet de mesurer la valeur du service. Vous pouvez facilement calculer les économies générées pour chaque instance de réutilisation, notamment le temps gagné dans l'architecture, la conception, le développement et le test.

En outre, la réutilisation implique un nombre moindre de services à gérer et à détecter. La réutilisation permet donc de réaliser des économies, et la fréquence d'utilisation de générer de la valeur.

Quelques concepts de base

Concevoir une architecture SOA peut s'avérer difficile, voire cauchemardesque. Les idées qui parfois semblent bonnes sur le papier ne le sont pas nécessairement dans la pratique. Un grand nombre d'entre elles sont réellement à éviter. Cependant, comme le démontre ce guide, vous pouvez apprendre beaucoup des erreurs des autres, notamment :

- Pourquoi le taux de réutilisation, et non le nombre de services Web, est-il le véritable indicateur d'une bonne architecture SOA
- Pourquoi le respect des standards SOAP est-il primordial
- Quand mettre en place une architecture SOA, et quels en sont les avantages
- Comment mettre en place une architecture SOA en ajoutant une surcouche aux applications existantes
- Pourquoi un pare-feu XML ne convient-il pas dans une architecture SOA
- Pourquoi est-ce important de bien comprendre les objectifs et les avantages d'une architecture SOA avant de se lancer
- Quelles règles métier votre stratégie SOA doit-elle concerner
- Pourquoi la SOA ne nécessite pas une approche de type "on efface tout et on recommence"
- Pourquoi le clonage et la reconfiguration des applications n'est pas la bonne méthode
- Pourquoi devez-vous être prudent quant au partage de votre WSDL
- Pourquoi vos partenaires sont vos meilleurs alliés lors de la sécurisation de votre validation de schéma

Et plus important encore, vous acquerrez la sagesse nécessaire pour prendre les bonnes décisions lors du lancement de votre initiative SOA.

A propos de Actional et Progress Software

La gouvernance des systèmes avec Actional de Progress Software

Actional est la première plate-forme de gestion des systèmes d'information SOA ou traditionnels, qui ne se limite pas à une approche "service par service". Elle offre une vision holistique des systèmes, une gouvernance dynamique et complète et une optimisation des processus ; ceci permet aux entreprises de créer des "cartes de flux d'activité" temps réel. Ces cartes sont créées à partir de transactions réelles et indiquent comment chaque processus utilise les services et l'infrastructure qui les supporte.

Actional permet de visualiser l'ensemble des processus exécutant des transactions au niveau le plus fin, de gérer l'architecture SOA du point de vue des processus, et aussi de contrôler l'application du code de gouvernance.

Actional 6.0 permet également aux collaborateurs de personnaliser un tableau de bord BAM (Business Activity Monitoring) afin d'y reporter des indicateurs stratégiques (KBI : Key Business Indicators), tels que le nombre d'unités expédiées, le délai entre la prise de commande et le recouvrement de la facture, ou le volume moyen des commandes… Ces indicateurs stratégiques sont directement basés sur les processus en cours d'exécution.
Actional 6.0 inclut également une fonctionnalité de gouvernance d'exécution automatisée (ou ARG pour Automated Runtime Governance) permettant de détecter les services et processus et d'appliquer automatiquement des politiques définies, ainsi qu'une fonctionnalité de sécurité "Trust Zones" empêchant toute utilisation non autorisée ou non conforme des services, et détournant ainsi les attaques potentielles à l'intérieur du réseau SOA. Actional fournit également une nouvelle interface graphique Web, la prise en charge avancée des standards, et une gamme étendue de plates-formes prises en charge.

Actional 6.0 résout un problème fondamental auquel la quasi-totalité des déploiements SOA et non SOA sont confrontés : des services et processus corrompus susceptibles d'impliquer des risques opérationnels et de sécurité. Actional résoud ce problème en détectant automatiquement les transactions, les services, notamment les services corrompus, dès leur déploiement, et en appliquant automatiquement les politiques informatiques, internes et de sécurité en vigueur. Cela permet aux entreprises de réduire de manière significative les risques associés aux services et processus corrompus, et de les aligner dynamiquement avec les politiques définies.

A propos de Progress Software

La suite SOA de Actional est un pionnier dans la gestion de services Web et fournit la visibilité, la sécurité, et le contrôle des activités de services et des processus dans un environnement d'exécution. La suite Actional surveille et contrôle les transactions asynchrones de longue durée lorsqu'elles transitent sur une vaste gamme de technologies (par exemple, SOAP/WS-*, HTTP, .NET, JMS, JDBC™, RMI et EJB, des serveurs d'applications tels que WebLogic, WebSphere et JBoss, des réseaux de type Cisco AON, Reactivity et Layer 7, ainsi que de nombreuses marques de bus ESB, dont Sonic ESB® et BEA AquaLogic Service Bus).
Actional est une gamme de produits de Progress Software Corporation (Nasdaq : PRGS), l'un des leaders mondiaux des logiciels avec plus de 447 millions de dollars de chiffre d'affaires.
Progress Software Corporation (Nasdaq : PRGS) est un fournisseur mondial des logiciels d'infrastructure SOA couvrant tous les aspects du développement, du déploiement, de l'intégration et de la gestion d'applications professionnelles. Son siège social se trouve à Bedford (Massachusetts).

Pour plus d'informations :

www.progress.com
Téléphone : +33.1.41.16.16.00
E-mail : frmkg@progress.com