

Livre Blanc Comprendre SCA

Obeo - S.A.R.L au capital de 45 000 € - 485 129 860 RCS NANTES 2 rue Robert Schuman 44400 REZE Siret : 485 129 860 00022 N° intracom : FR40485129860	Tel : 02 51 13 51 42 Email : info@obeo.fr www.obeo.fr
--	---

Résumé : *L'objectif de ce livre blanc est de montrer les avantages que l'on peut tirer des architectures orientées services (SOA) et notamment de l'apport des spécifications Service Component Architecture (SCA) pour construire des applications s'inscrivant dans une SOA.*

Table des mises à jour			
Version	Date	Auteur(s)	Mises à jour
v1.0	23/01/09	Stéphane Drapeau Maxime Porhel Etienne Juliot	Version initiale

Table des matières

1 - Service Oriented Architecture (SOA).....	3
1.1 - Le service.....	3
1.2 - Le bus de services.....	3
1.3 - Avantages.....	4
2 - Service Component Architecture (SCA).....	5
2.1 - Quelle pérennité ?.....	5
2.2 - A quels objectifs techniques SCA répond ?.....	5
3 - Les gains apportés par SCA.....	6
3.1 - Comment faciliter la réutilisation au sein du SI ?.....	6
3.2 - Comment faciliter l'interconnexion au sein du SI et avec d'autres SI ?.....	6
3.3 - Comment fiabiliser le SI et les communications sur le bus ?.....	7
4 - Les plate-forme d'exécution SCA.....	8
4.1 - Implémentations open source.....	8
4.2 - Implémentations payantes.....	8
5 - Eclipse SCA : le projet SCA Tools.....	9
6 - Success Stories.....	10
6.1 - Gartner.....	10
6.2 - IBM.....	10
6.3 - Amdocs.....	10
7 - En conclusion.....	11

1 - Service Oriented Architecture (SOA)

Les systèmes d'information (SI) des entreprises sont habituellement constitués de diverses applications et données formant ce qu'on appelle son **héritage**. Avec les fusions de groupe et l'évolution des technologies, cet héritage a une propension à devenir hétérogène et à se spécialiser par métier, provoquant un fonctionnement en **silos**. Cela conduit à un cloisonnement des différents métiers empêchant la transversalité et masquant une vision globale du système d'information de l'entreprise.

L'intégration des applications de l'entreprise (EAI) est une solution à ce problème. Elle consiste à développer des connecteurs permettant de faire communiquer entre-eux les différents silos de l'entreprise.

Ainsi, une **architecture orientée services** (SOA pour *Services Oriented Architecture*) est une architecture logicielle s'appuyant sur un ensemble de services simples. L'objectif d'une architecture orientée services est de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées **services**, fournies par des composants et de décrire finement le schéma d'interaction entre ces services.

L'idée sous-jacente est de cesser de construire la vie de l'entreprise autour d'applications mais de construire une architecture logicielle globale décomposées en services correspondant aux processus métiers de l'entreprise.

1.1 - Le service

Le service est un composant clef du SOA. C'est une fonction ou fonctionnalité bien définie. C'est aussi un composant autonome qui ne dépend d'aucun contexte ou service externe.

Le SOA consiste en une collection de services qui interagissent et communiquent entre eux. Cette communication peut consister en un simple retour de données ou en une activité (coordination de plusieurs services).

Un service est une entité de traitement qui respecte les caractéristiques suivantes :

- **Large Granularité** (*coarse-grained*) : Les opérations proposées par un service encapsulent plusieurs fonctions et opèrent sur un périmètre de données large au contraire de la notion de composant technique.
- **Interface** : Un service peut implémenter plusieurs interfaces, et différents services peuvent implémenter une interface commune.
- **Localisable** : Avant d'appeler un service, il est nécessaire de le trouver.
- **Instance unique** : A la différence des composants qui sont instanciés à la demande et peuvent avoir plusieurs instances, un service est unique.
- **Couplage faible** (*loosely-coupled*) : Les services sont connectés aux clients et autres services via des standards. Ces standards assurent le découplage, c'est-à-dire la réduction des dépendances. Ces standards sont des documents XML comme dans les web services.

1.2 - Le bus de services

L'implantation la plus commune de SOA est celle basée sur un bus de services. Ce bus a un rôle de médiateur (*middleware*) entre le consommateur et le producteur du service, il permet de réaliser le couplage lâche. Le bus peut aussi fournir différents services :

- fractionnement, combinaison, etc. permettant de construire l'appel sur plusieurs services,
- gestion de version de service,
- supervision et contrôle des services.

On parle généralement d'ESB (Enterprise Service Bus), outil de nouvelle génération pour l'EAI construit sur des standards comme XML, JMS ou encore les services web. La différence majeure avec l'EAI réside dans le fait que l'ESB propose une intégration complètement distribuée évitant ainsi un goulet d'étranglement sur le bus de messages et offrant une meilleure robustesse (figure 1).

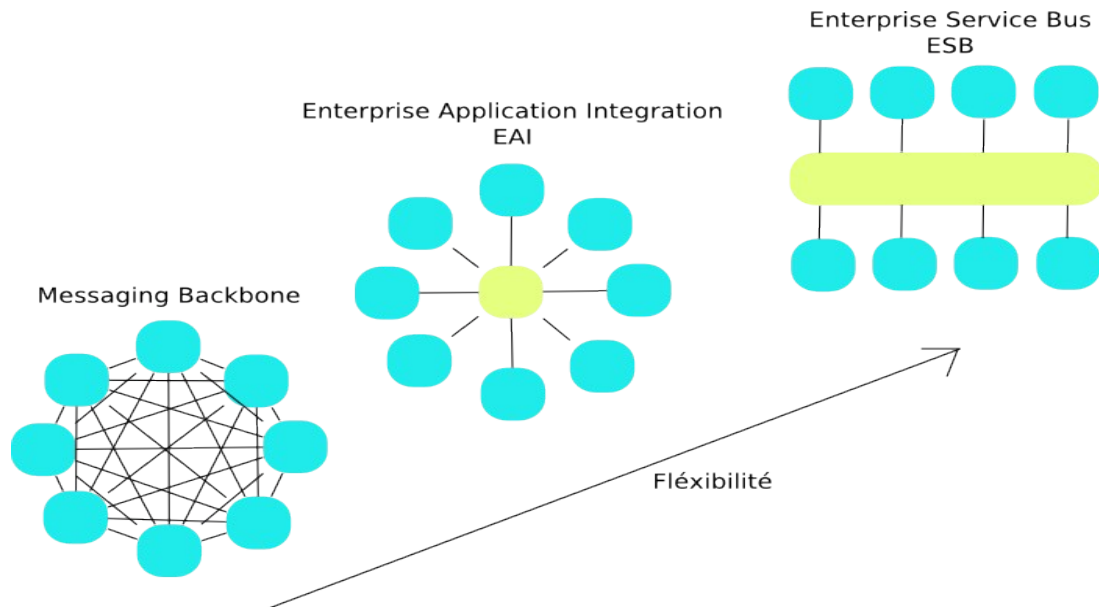


Figure 1: Historique de l'intégration (Tiré du Livre Blanc Xebia - Comprendre et savoir utiliser un ESB dans une SOA)

Ces "mini-serveurs" contiennent la logique d'intégration et peuvent être déposés n'importe où sur le réseau. L'ESB est également un outil d'échange asynchrone disposant d'interfaces standardisées (SOAP, JMS,...) ou intégrées (JBI). Il peut aussi offrir des services à valeur ajoutée (traduction, transformation...) activés par des événements ou des batches.

1.3 - Avantages

Une architecture orientée services permet d'obtenir de nombreux avantages :

- Une modularité permettant de **remplacer** facilement un composant (service) par un autre.
- Une **réutilisabilité** possible des composants (par opposition à une système tout-en-un fait sur mesure).
- De meilleures possibilités d'**évolution** (agilité et flexibilité) : il suffit de faire évoluer un service ou d'ajouter un nouveau service.
- Une interconnexion des applications facilitée par l'emploi de **standards**.
- Une plus grande **tolérance aux pannes**.
- Une **maintenance** facilitée.
- Une **unicité** des règles de gestion et des données.
- D'**aligner le SI sur le métier de l'entreprise**.

SOA n'est pas une technologie, ni une recette, encore moins un produit. C'est une façon de penser et de concevoir le Système d'Informations. SCA propose un modèle de composants permettant de construire une architecture SOA. La plus value majeure de SCA est de **ne pas imposer de choix technologiques** : SCA permet de décrire un ensemble de services et leurs interactions indépendamment des technologies utilisées.

2 - Service Component Architecture (SCA)

2.1 - Quelle pérennité ?

Les spécifications Service Component Architecture (SCA) sont issues du consortium Open SOA formé des principaux éditeurs de logiciels (IBM, SAP, Oracle, Iona, ...). Ces spécifications sont en cours de normalisation par un organisme internationalement reconnu : OASIS.

Tous les grands éditeurs, IBM en tête, ont annoncé que SCA est au cœur de leurs offres de Bus logiciels.

2.2 - A quels objectifs techniques SCA répond ?

SCA vise à simplifier la construction d'architectures orientées services (SOA) en adressant plus particulièrement :

- La composition : comment packager un composant logiciel afin que d'autres applications puissent l'utiliser ?
- L'assemblage : comment les composants peuvent fonctionner ensemble ?
- Politique : comment associer des politiques non fonctionnels (restriction d'accès, signature numérique, etc.) aux composants ?

Avec les techniques classiques, toutes ces tâches sont de plus en plus difficiles à mettre en œuvre : il existe de nombreuses technologies de description d'interfaces métier, de protocoles de communication et une multitude de langages de programmation. De plus, dans un Système d'Information classique, des anciennes technologies côtoient des nouvelles technologies et des ERP, et tous doivent pour autant communiquer ensemble.

Cette complexité rend difficile la construction d'une architecture orientée services flexible répondant aux besoins de l'entreprise. SCA fournit une solution unique cachant les détails sous-jacents et simplifiant le développement. [Introducing SCA](#) propose une bonne introduction à SCA.

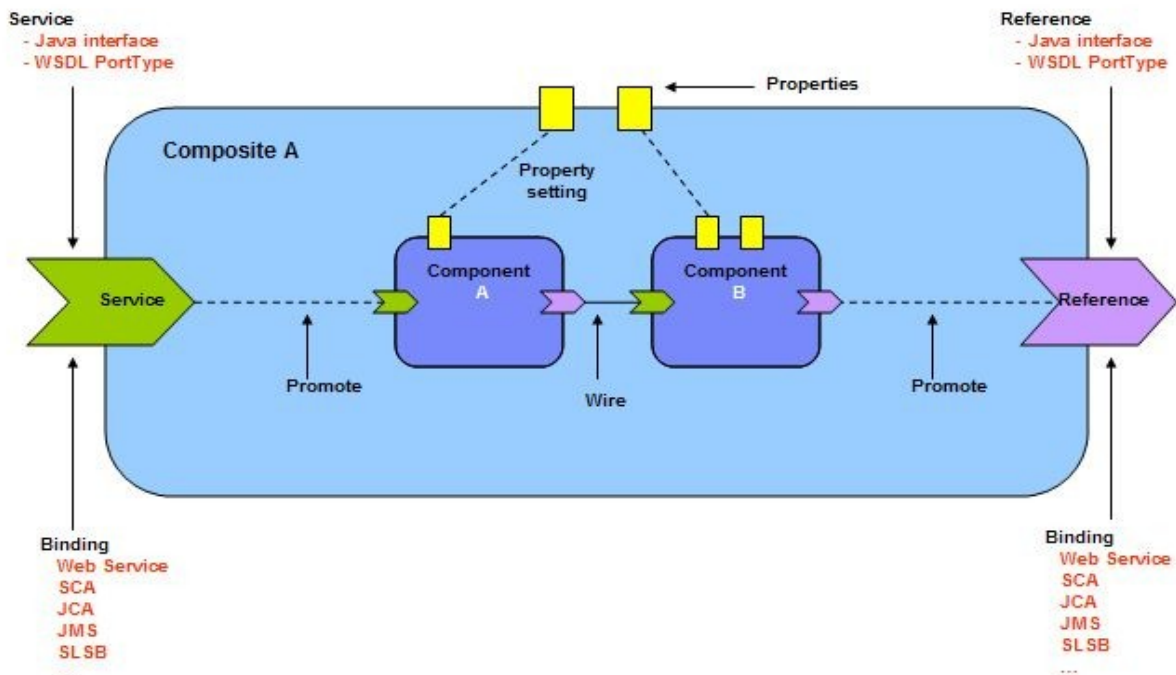


Figure 2 : exemple de diagramme SCA

3 - Les gains apportés par SCA

Les spécifications SCA se composent de 4 parties majeures ; nous les présentons dans la suite afin de démontrer que SCA simplifie le travail des développeurs, des administrateurs et des architectes.

3.1 - Comment faciliter la réutilisation au sein du SI ?

3.1.1 - Solution : Assemblage SCA

L'assemblage définit comment un nouveau composant ou une application existante expose des services métiers, et comment ces services sont assemblés en « composites » pour proposer un découpage en fonctionnalités alignées sur l'organisation de l'entreprise .

Ainsi, un composite est un groupe de composants accomplissant une tâche particulière. La notion de composite simplifie la réutilisation en exposant un groupe de composants comme un seul service.

Les spécifications SCA d'assemblage décrivent également comment lier les composants et les composites. Accéder un service peut nécessiter du chiffrement, une authentification, ou l'utilisation d'une queue de messages. SCA permet de définir ces aspects en dehors du code du service, permettant ainsi de les modifier/changer sans impact sur le code.

Business value : SCA rend la réutilisation de briques existante plus simple.

Si une fonction métier est implémentée comme un composite SCA, toutes les applications ayant besoin de cette fonction métier peuvent simplement réutiliser ce composite. SCA permet également de définir des propriétés pour un service sans changer l'implémentation de ce service.

3.2 - Comment faciliter l'interconnexion au sein du SI et avec d'autres SI ?

3.2.1 - Solution #1 : Modèle client et modèle d'implémentation SCA

Les modèles Client et d'implémentation définissent comment les services sont accédés par divers langages de programmation. Cela signifie que les développeurs ont seulement une interface à connaître, qu'ils peuvent l'implémenter et l'appeler dans la technologie de leur choix et que celle-ci peut être utilisée avec n'importe quel composant compatible SCA. Actuellement, OpenSOA fournit les spécifications pour Java, Spring, BPEL, C++, Cobol, C et Java EE. Cette liste ne cesse de s'étendre et des implémentations propriétaires existent pour d'autres technologies.

3.2.2 - Solution #2 : Modèle de liaison SCA

Les liaisons spécifient les mécanismes permettant d'accéder aux composants. Les applications construites avec SCA peuvent utiliser les services web, JMS, JCA ou les EJB. L'intérêt de SCA est que ces méthodes de liaison sont définies en dehors du code métier.

Business value : SCA facilite l'intégration

- SCA donne aux développeurs une interface unique pour des services écrits dans divers langages et architectures. Une interface unique signifie moins de temps d'apprentissage et plus de temps à consacrer au développement de code métier. De plus, la mise en place de pont entre ERP et des SI de partenaires de l'entreprise se trouve très largement facilitée car elle n'impose pas de choix technologique.
- Les méthodes de liaison fournies par SCA permettent aux développeurs d'utiliser un grand nombre de services sans connaître en détails comment y accéder. En définissant ces méthodes d'accès en dehors du code, elles peuvent être modifiées/remplacées sans impact sur le code.

3.3 - Comment fiabiliser le SI et les communications sur le bus ?

3.3.1 - Solution : politiques de qualité de service SCA

Le framework de politiques SCA permet de définir des politiques pour la sécurité, l'authentification, la qualité de service et d'autres aspects important pour les services. Le framework utilise les standards ouverts WS-Policy et WS-PolicyFramework pour décrire ces politiques.

Un administrateur peut préciser que les données envoyées à un service sont cryptées, que l'utilisateur d'un service doit être authentifié avant d'accéder au service, ... Comme tous les aspects de SCA, ces politiques peuvent être définies en dehors du code du service. Encore une fois, cela signifie qu'un changement de politique ne nécessite pas la modification du service.

Business value : SCA permet de fiabiliser simplement le SI

La possibilité de définir les politiques en dehors des services permet aux administrateurs d'associer et de modifier des politiques sans modification sur le code. De plus, cela permet aux administrateurs d'avoir une vue d'ensemble des politiques utilisées.

4 - Les plate-forme d'exécution SCA

4.1 - Implémentations open source

Il existe plusieurs implémentations open source des spécifications SCA. Les principales sont présentées dans cette section.

Le projet [Apache Tuscany](#) est une implémentation open source (licence Apache 2.0) des spécifications SCA. Tuscany implémente également les spécifications Service Data Objects (SDO) qui simplifient et unifient l'accès et la manipulation des données provenant de sources de données hétérogènes (bases de données relationnelles, sources de données XML, services web, EIS, etc). Tuscany permet la définition de nouvelles extensions pour de nouveaux types d'interface, d'implémentation et de méthodes de liaison, et repose sur le sérieux reconnu de la fondation Apache (déjà à l'origine de Tomcat, du serveur web Apache, Axis, Ant, ...). En plus de ces API d'extension, Tuscany peut également être étendu en misant sur son architecture modulaire et le fait que son code source soit ouvert.

Le projet collaboratif [SCOrWare](#), financé par l'Agence Nationale de Recherche (ANR) a pour ambition de fournir une implémentation en logiciels libre des spécifications SCA. Celle-ci comprend le runtime Frascati et l'outillage d'aide à la conception et au développement d'applications SCA (cet outillage est décrit dans la section suivante). La plate-forme d'exécution **Frascati** est construite au dessus du modèle de composants Fractal du consortium OW2. Elle offre des fonctionnalités avancées telles que la reconfiguration dynamique des assemblages SCA, une usine à liaisons supportant différents protocoles de communication inter-composants, un service de transactions, un service de courtage sémantique des composants SCA, une machine de déploiement d'architectures SCA et des consoles graphiques d'administration.

Le projet [Fabric3](#) est une autre implémentation sous licence Apache. Il offre la possibilité de déployer et gérer des services dans des environnements distribués. Fabric3 est compatible avec de nombreux outils de développement et technologies populaires (JEE, Spring,...). Son architecture modulaire permet de s'ajuster aux technologies requises sur un projet. Il peut être utilisé pour construire, déployer et gérer des application SCA au sein de divers middleware (serveur d'application JEE, conteneurs de Servlet, conteneurs OSGi).

Le projet [Newton](#) est un framework OSGi distribué qui permet l'instanciation et la gestion d'applications SCA au sein d'environnements d'entreprise. Newton est capable de déployer et de gérer des assemblages de composants distribués par des ajustements dynamiques en fonction des erreurs et des changements de topologie des réseaux.

4.2 - Implémentations payantes

Il existe aussi des implémentations payantes, telles que :

– [IBM WebSphere Feature Pack for SOA](#) qui permet d'ajouter la prise en charge des applications SCA au sein du serveur d'application IBM WebSphere.

– [Rogue Wave® HydraSCA](#) qui permet aux développeurs de construire des composants individuels puis de les lier au sein de « business process » agiles et de les exécuter de manière concurrente.

– [Oracle SALT](#) (Service Architecture Leveraging Tuxedo) qui permet d'adapter le moniteur transactionnel Oracle Tuxedo au SOA. La dernière version (10gR3) de l'outil SALT supporte les spécifications SCA. Cet outil fait partie de la famille Oracle Fusion Middleware.

5 - Eclipse SCA : le projet SCA Tools

[SCA Tools](#) est une suite d'outils permettant de faciliter le développement d'applications SCA. C'est un projet faisant partie du projet SOA Tools Platform (STP) de la fondation Eclipse. Ces outils sont issus du projet collaboratif SCOrWare.

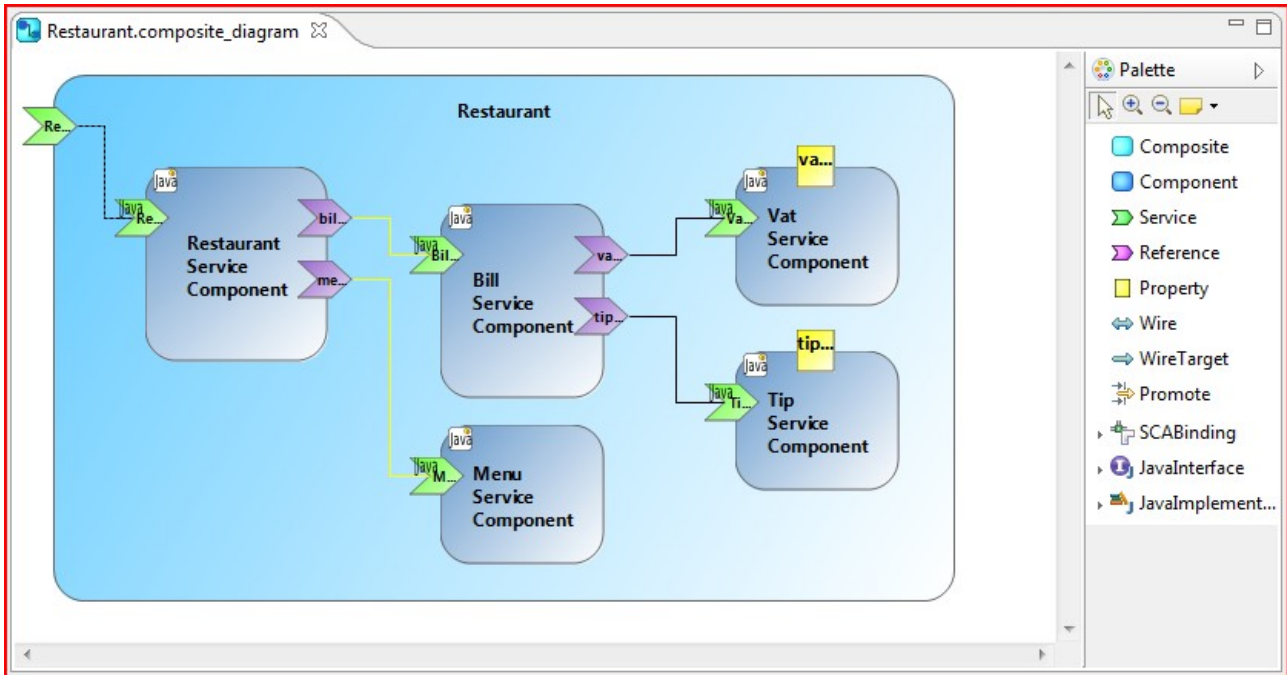


Figure 3 : diagramme SCA dans l'éditeur graphique de SCA Tools

SCA Tools est un environnement de développement open source qui repose sur les dernières spécifications proposées par le consortium Open SOA.

Il propose les fonctionnalités suivantes:

- Éditeur graphique pour la construction d'assemblages SCA,
- Éditeurs xml et forms pour la construction d'assemblage SCA,
- Extensibilité de l'éditeur graphique pour supporter différents moteurs d'exécution,
- Validation des assemblages SCA,
- Compatibilité avec les principaux moteurs d'exécution open source : Frascati, Tuscany, Newton et Fabric3.

D'autres fonctionnalités sont en cours de développement :

- Introspection de code Java permettant de construire automatiquement des composants SCA,
- Éditeur de politiques non fonctionnelles SCA,
- Intégration avec des services de courtage.

6 - Success Stories

6.1 - Gartner

Selon un rapport de Gartner datant de Mars 2006, titré « SCA est un Gagnant de la Quête destinée à établir une notation commune pour la SOA », Jess Thompson, vice-président de Gartner Research, déclarait que *"L'un des plus importants aspects de la SCA est qu'elle établit une base pour une notation standard capable d'exprimer une série de concepts qui précise l'architecture orientée services."*

6.2 - IBM

IBM a fait le choix pour sa suite d'outils commerciaux de bus de services IBM WebSphere Application Server de faire le choix de mettre SCA en son coeur et de le promouvoir intensément.

Or, ce sont les équipes d'IBM qui sont à l'origine du projet Apache Tuscany. A travers des contacts privilégiés avec Obeo, plusieurs retours d'expérience positifs sur des SI importants du monde bancaire américain permettent de valider la robustesse du serveur OpenSource. De plus, en le plaçant au coeur de l'offre commerciale, on fait aisément être rassurer sur sa maintenance sur le long terme, tout comme l'a été Tomcat ou Axis au sein de WebSphere.

6.3 - Amdocs

Amdocs is the leading supplier of customer experience systems service providers worldwide. Within Amdocs, the OSS Division has adopted Tuscany as the SCA implementation at the heart of its solution for delivery of SOA services for Cramer6 OSS Suite.

The current release of the Cramer6 OSS Suite provide both out-of-the-box SOA services as well as tools allowing the creation of new OSS-specific SOA services. These services are built on top of the Tuscany runtime and allow Amdocs customers to integrate their existing systems with Cramer6 OSS Suite services using whatever technology is appropriate. Thus, customers may access the services through any appropriate transport mechanism and from any "native" or SCA based client. They can also publish new services that can be accessed over a variety of different transport mechanisms with minimal recoding.

"SCA has helped us to widen our support for SOA by giving us a mechanism to create SOA Services" said DAVE ETTLE, Senior Vice President, Products & Technology "We believe Tuscany provides the most appropriate SCA runtime implementation for our needs".

Amdocs intends to continue to use SCA/Tuscany for its OSS applications.

Vous trouverez également des retours d'utilisateurs de Tuscany à l'adresse suivante : <http://tuscany.apache.org/projects-using-tuscany.html>

7 - En conclusion

SCA vous permet de :

- **Économiser du temps et de l'argent** en permettant au développeur de se consacrer à la logique métier :
 - Une API simple signifie un apprentissage plus court avec moins d'erreurs et plus de temps consacré à la logique métier. SCA vise à abstraire le modèle de programmation du middleware de la logique métier et ainsi réduire la complexité sous-jacente.
 - De plus, les services sont plus faciles à développer car la sémantique de chaque service est moins complexe à appréhender que l'ensemble des services composant l'application finale.
 - Chaque composant peut également être développé par différentes équipes sans se préoccuper des détails d'implémentation des autres équipes.
- **Encourager la réutilisation** :
 - Les développeurs créent des composites proposant des services clairement identifiés par des interfaces. Cela permet de développer, tester et déboguer chaque composant indépendamment des autres et de les réutiliser.
 - SCA permet également d'intégrer facilement des développements non SCA existants, préservant ainsi les investissements faits et un passage progressif à SCA.
- **Gérer votre système** :
 - Les politiques étant définies en dehors du code réalisant le service, vous pouvez modifier ces politiques sans impacts sur le code.
 - La vision graphique de l'assemblage SCA dans un diagramme permet de donner une vision globale du SI aux architectes et au DSI.
- **Être indépendant des technologies** : langages de programmation, méthodes d'accès, protocoles d'interopérabilité. Contrairement à d'autres approches, Spring par exemple, SCA supporte une grande variété de langage de programmation pour l'implémentation et la définition des interfaces des composants permettant ainsi l'indépendance. SCA ne propose pas un nouveau langage de programmation, une nouvelle technologie d'accès aux services mais permet d'assembler ces technologies existantes.
- **Déployer, modifier et remplacer dynamiquement votre code** : SCA permet de déployer dynamiquement des composants afin de satisfaire rapidement aux besoins métiers.