

## SensorBean : Un modèle à composants pour les services basés capteurs

Mikael Desertot, Cristina Marin, Didier Donsez

Federation IMAG, Laboratoire LSR, Equipe Adèle,  
220 rue de la Chimie, Domaine Universitaire, BP 53  
38041 Saint Martin d'Hères - France  
Mikael.Desertot,Cristina.Marin,Didier.Donzes@imag.fr

---

### Résumé

Les services basés capteurs (SBC) proposent de collecter, contrôler, analyser, accéder et réagir aux données des capteurs comme des informations RFID, une position GPS, une puissance consommée, une température. Ces services sont distribués sur un ensemble de machines et de plates-formes hétérogènes. La complexité de mise en œuvre de tels services requière des outils d'ingénierie logicielle pour soulager l'architecte et le développeur qui sont souvent des experts métier et non pas des experts des technologies nécessaires. Notre proposition porte sur la définition d'un modèle à composant dédié au développement de SBC, appelé SensorBean. SensorBean se différencie des modèles à composants usuels par son orientation services dynamiques et par l'introduction de connecteurs suivant le patron producteur-consommateur adaptés au traitement de flots de mesures. Cette proposition a été en partie validée pour un démonstrateur couplant OSGi et J2EE et un premier prototype de conteneurs pour la partie OSGi de l'implantation du modèle.

**Mots-clés :** Composants, Conteneurs, Services orientés capteurs, OSGi, J2EE.

---

### 1. Introduction

Le développement des capteurs communicants dans de nombreux domaines industriels (industrie manufacturière, domotique/immotique, transport) et la généralisation d'Internet offrant des possibilités de connexions filaires ou sans-fil, simples et à bas coût (bus de terrain, CAN, WiFi, Bluetooth, ZigBee, RFID, GPRS) favorisent la création de nouveaux services à forte valeur ajoutée basés sur l'exploitation des mesures acquises par les capteurs.

Ces **services basés capteurs**(SBC) permettent à une entreprise ou organisation d'intégrer dans son système d'informations des données récoltées par des milliers des capteurs hétérogènes disséminés dans l'environnement surveillé. Ces services sont utilisés pour acquérir, collecter, filtrer, agréger, analyser, sauvegarder et réagir aux mesures acquises. Ces mesures peuvent être aussi variées que l'identifiant RFID d'un colis en transit lu par un lecteur, la position GPS d'un véhicule, la puissance électrique consommée par une chaudière, une image infrarouge d'une caméra de surveillance. Les informations remontées peuvent ensuite être utilisées pour améliorer le pilotage des équipements ou pour auditer le fonctionnement des systèmes surveillés.

Au cours du temps, les architectures centralisées dans lesquelles un serveur collectait des mesures directement sur les capteurs ont montré leurs faiblesses par rapport au caractère dynamique de l'environnement en cours de production. Aujourd'hui, de nouvelles architectures sont envisagées pour satisfaire les besoins d'exploitation à distance d'équipements : l'utilisation de passerelles Internet intelligentes et normalisées entre les équipements industriels et les serveurs IT. L'architecture assez complexe et parfois le caractère critique de l'application utilisant les SBC obligent à prévoir, dès la conception des services basés capteurs, les changements induits dans l'architecture de l'application par la dynamique de l'environnement. Des situations comme le branchement, le retrait ou la reconfiguration de nouveaux équipements, la défaillance transitoire ou définitive d'un capteur ou bien encore le redémarrage d'une passerelle doivent être gérés d'une manière la plus automatique et autonome possible.

La mise en œuvre de services basés capteurs requière donc les outils d'ingénierie logicielle permettant la mise en place rapide et sûre de ces services. Ces outils doivent néanmoins permettre de récupérer tous les développements patrimoniaux déjà réalisés.

L'approche à composants[1] a fait ses preuves pour la construction d'applications complexes. Cependant les principaux modèles à composants industriels sont destinés plutôt aux applications IT d'entreprise(EJB, CCM, .NET). D'autres modèles (PECOS[2], ROBOCOP[3]) ne ciblent que les applications pour des systèmes contraints ou embarqués, ce qui rend inadéquate leur utilisation pour développer des services basés capteurs résidant à n'importe quel niveau de l'architecture mentionnée. Généralement, ces modèles à composants n'offrent pas des mécanismes adéquates aux traitements de flots de mesures autour desquels les services basés capteurs sont articulés. De même, il reste difficile de prendre en compte pour l'instant les composants patrimoniaux même si quelques propositions existent sur ce point.

Il nous est donc apparu nécessaire de définir un modèle à composants spécialisé au développement des services basés capteurs. Notre proposition porte sur la définition d'un nouveau modèle à composants distribué et dynamique, appelé SensorBean. Le modèle défini a été validé par un premier prototype qui s'appuie à la fois sur des modèles à composants concrets dédiés plate-formes (EJB pour un serveur J2EE, Fractal pour une plate-forme OSGi[4]) et sur des intergiciels standards dédiés au traitement des flots de mesures acquises par les capteurs (WireAdmin OSGi, OMG DDS, NIST IEEE 1451.1).

La suite de l'article est organisée de la manière suivante : nous présenterons notre proposition de modèle à composants SensorBean dédié aux services bases capteurs. La section 3 présente un démonstrateur s'exécutant sur des passerelles OSGi et J2EE qui a été le point de départ de la réflexion pour un nouveau modèle à composants. Dans la section 4 nous présentons les travaux relatifs. Nous concluons enfin avec les perspectives et les conclusions de ce travail.

## **2. Modèle à composants SensorBean**

Destiné à faciliter le développement et le déploiement des services basés capteurs, le modèle à composants que nous proposons prend en considération tous les types d'interactions susceptibles d'apparaître dans le contexte présenté. Comme les modèles à composants les plus courants, notre modèle offre des liaisons de type appel de méthodes (que l'on retrouve dans EJB, Fractal, CCM) et de type événementiel (JavaBeans, CCM). Il ajoute également une nouvelle liaison de type producteur-consommateur de flots de mesures, bien adaptée aux SBC. L'approche abordée est celle d'un modèle à composants orienté services dynamiques pour pouvoir prendre en compte le caractère fluctuant de l'environnement. Les applications sont composées d'assemblages de services basés capteurs dynamiques packagés dans des composants SensorBean.

### **2.1. Description du modèle à composants**

Cette section décrit le modèle à composants(figure 1) pour les SBC que cet article propose. Ce modèle est utilisé pour encapsuler des services basés capteurs et pour pouvoir composer ces services afin d'offrir la fonctionnalité d'une application basée SBC.

Nous avons identifié trois types possibles d'interaction entre les composants. En plus des types connus (appel de méthodes, événementiel), notre modèle à composants ajoute un troisième : interaction par flots de mesures capturées. Dans une application utilisant des services basés capteurs, ce dernier type de connexion entre composants est le plus important. Par rapport au modèle à composants SOFA[5], qui possède déjà un type de connexion par des flots d'octets, notre modèle à composants permet une liaison entre deux composants par flot de données typées, mis en pratique par un patron de communication producteur-consommateur.

#### **2.1.1. Facette et réceptacle**

Ces 2 ports sont utilisés pour les interactions de type appel synchrone de méthode. La connexion d'un réceptacle vers une facette est utilisée quand un composant(client) nécessite des services offerts par un autre composant(serveur). La facette est exprimée sous la forme d'une ou plusieurs interfaces Java qualifiées par des propriétés de courtage (au moment de l'assemblage). Le réceptacle est exprimé sous la forme d'une ou plusieurs interfaces Java restreintes par une expression de sélection sur les propriétés de courtage des facettes. L'établissement (resp. la rupture) des liaisons est réalisé par la plateforme lors

de l'instanciation du composant possédant le réceptacle ou lors de l'apparition (resp. disparition) des nouvelles facettes vérifiant l'expression de sélection.

### 2.1.2. Source et puit d'événements

Ces ports sont utilisés pour publier et souscrire des événements. La source et le puit d'événements sont exprimés en terme d'une ou plusieurs interfaces Java dont toutes les méthodes ne retournent aucune valeur (c.a.d void). Ils sont utilisés par exemple pour transmettre des événements représentant des informations sémantiques plutôt que des mesures brutes, comme par exemple la détection de mouvement dans une pièce ou le franchissement d'un seuil de surchauffe d'un four.

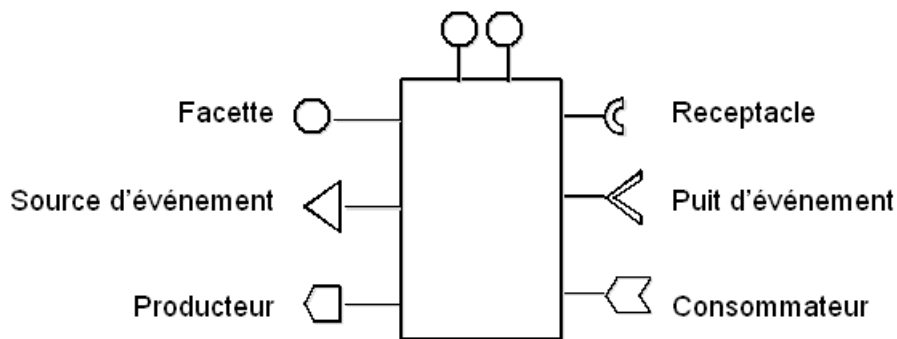


FIG. 1 – Le modèle à composants

### 2.1.3. Producteur et consommateur de flots de mesures

Ces ports sont principalement utilisés pour remonter des mesures brutes. Le producteur et le consommateur sont caractérisés par les types de données qu'ils sont capables de produire ou de consommer. Conformément au patron de conception Producteur-Consommateur, l'initiative de l'échange de données est partagée de manière symétrique par les deux ports, contrairement aux types de ports précédents. Un producteur disposant d'une nouvelle mesure capturée peut la diffuser vers des consommateurs connectés. De même, un consommateur peut demander la dernière valeur mesurée au producteur. Généralement, les producteurs représentent des capteurs et les consommateurs des actionneurs.

Chaque composant possède un certain nombre de propriétés non-fonctionnelles de base comme la persistance, le cycle de vie, la réactivité (période de déclenchement d'un thread de contrôle) et la sécurité. Ces propriétés de base peuvent être étendues en fonction de la plate-forme d'exécution sur laquelle est déployée le composant (transaction, concurrence sur J2EE et historique circulaire sur OSGi).

## 2.2. Architecture

La figure 2 présente une instance d'une application composée de plusieurs composants SensorBean. Pour automatiser les connexions entre les ports des composants, l'architecture d'une application est décrite par un ADL (Architecture Description Language). L'ADL décrit l'application en terme de connexions possibles entre les ports des composants au moyen des propriétés et expressions de courtage.

L'approche orientée service du modèle autorise la substitution de tout composant par un autre au cours de l'exécution si celui vérifie le contrat de service. Ainsi les connexions sont établies et validées dynamiquement quand les services requis par le composant sont disponibles sur la plate-forme d'exécution ou qu'un composant producteur d'un type de mesure est enregistré. Les propriétés de courtage associées permettent, par exemple, de sélectionner parmi un sous-ensemble de composants implémentant le même service nécessaire, celui qui correspond mieux aux critères du composant ou de garantir la com-

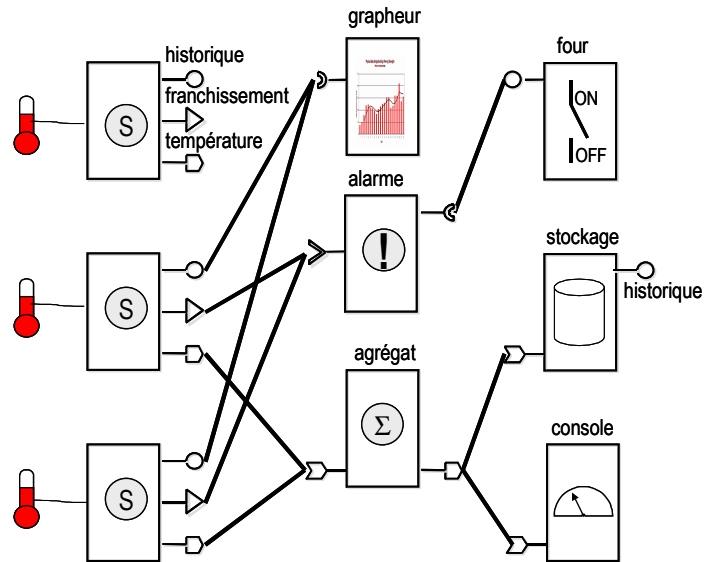


FIG. 2 – Instance d’application basée SBC

patibilité des données échangées entre un producteur ou un consommateur. Le courtage permet également de récupérer des composants patrimoniaux pourvus qu’ils respectent les interfaces de services. Cette approche reprends les principes de ServiceBinder[6] mais les étends aux connecteurs événementiels et producteur-consommateur.

### 3. Prototype

Le modèle à composant SensorBean est issu de la réflexion autour d’un service basé capteur s’appuyant sur deux grands standards que sont OSGi[1], un consortium proposant une spécification de plate-forme de déploiement de services logiciels basé sur une programmation orienté services dynamiques(POSD)[7], et J2EE. D’un coté, une passerelle OSGi(l’implémentation open source Oscar) gère une variété de capteurs réels (capteur de température, camera de surveillance, récepteur GPS). De l’autre coté, un serveur J2EE (l’implémentation open source JonAS) centralise dans une base de données les informations collectées sur les capteurs par la passerelle(figure 3) et peut éventuellement effectuer des traitements sur ces valeurs. Le serveur est également capable de piloter à distance la passerelle. Les échanges d’informations entre les deux plates-formes sont possibles en mode *push*(à l’initiative de la passerelle) et en mode *pull* (à l’initiative du serveur).

L’architecture déployée sur le serveur est basée sur des composants classique de J2EE. L’architecture de la passerelle s’appuie sur l’utilisation de ServiceBinder[6] pour automatiser la liaison entre les services et de WireAdminBinder, une extension du service WireAdmin déjà spécifié par OSGi, qui automatise la création des liens entre les producteurs de données (capteurs) et les consommateurs (services de monitoring ou de communication vers le serveur J2EE via SOAP, SMTP, JMS). Autour de cette architecture, nous avons définis des propriétés non fonctionnelles spécialisées par le coté passerelle. Celles-ci définissent le cycle de vie et la persistance des composants en fonction des arrêts (doux et brutal) et des redémarrages des composants, des unités de déploiement (les bundles OSGi) et de la passerelle.

Le modèle présenté dans la partie 2 nous permet de décrire l’architecture qui devra être projetée vers différentes plates-formes cibles, éventuellement hétérogènes. Si J2EE fournit un modèle à composants évolué, OSGi ne propose qu’un modèle à composants très basique (pas d’interconnexion possible vers

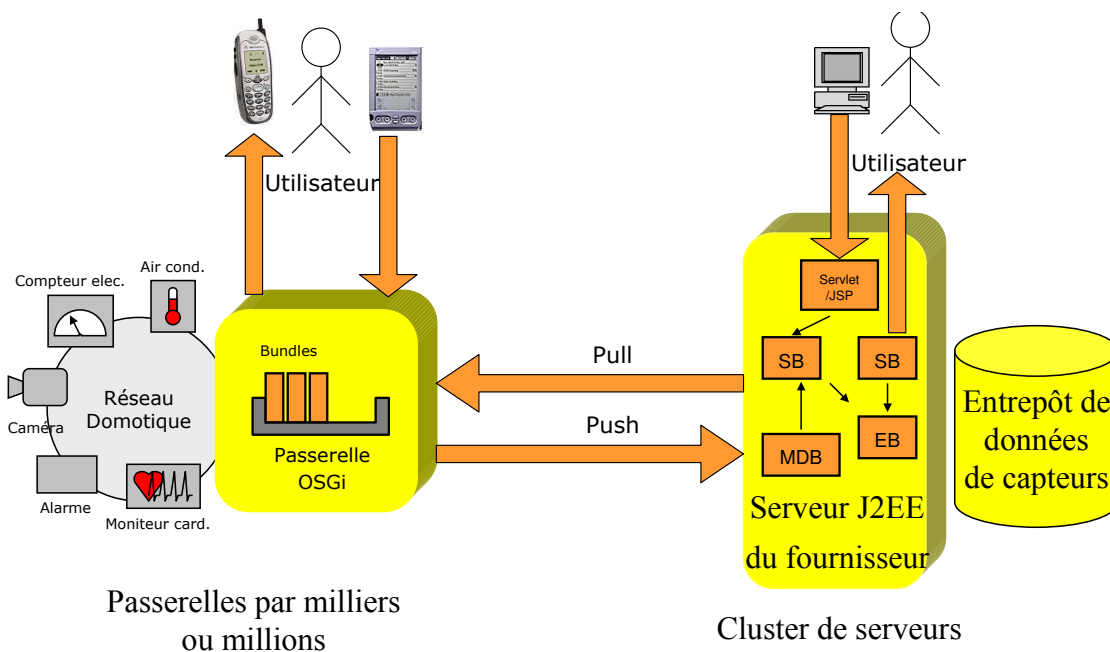


FIG. 3 – Prototype SensorBean

l'extérieur, pas de notion d'instance), ce qui limite les possibilités de projections à partir du modèle décrit. Pour lever ce verrou, une unité de déploiement OSGi embarque le conteneur du composant SensorBean. Le conteneur est réalisé pour fonctionner de manière autonome sur les plate-formes d'exécution. Ainsi, il embarque la partie de l'ADL de l'application qui concerne le composant dont il est responsable. Ainsi le composant peut être connecté et interagir avec des "composants" du modèle natif de la plate-forme d'hébergement (services pour OSGi, Enterprise Beans pour EJB).

L'expérimentation d'un tel conteneur a été réalisée en utilisant OSGi comme conteneur de déploiement pour le modèle à composant Fractal[8]. Cette implémentation complète le générateur de FROGi[9] qui a déjà démontré l'intérêt d'utiliser d'OSGi pour le déploiement de modèles à composants (comme Fractal).

#### 4. Travaux relatifs

Nous nous sommes intéressés aux modèles à composants se rapprochant le plus de SensorBean. Comme nous l'avons déjà dit, aucun des modèles à composants industriels (EJB, .NET, JavaBeans) ou ceux issus de la recherche (Fractal, CCM, Excalibur) n'offrent tous les types d'interactions nécessaires dans le contexte de notre travail. Ils prennent en compte seulement des connexions de types client-serveur ou événementiels (CCM). Notre attention s'est aussi portée sur le modèle à composants SOFA[5] qui prend en compte les connecteurs de type producteur-consommateur. Cependant ce type de connecteur est restreint à l'échange de données binaires non typées.

L'autre approche étudiée est celle des modèles à composants orienté services. Parmi les travaux étudiés, OpenWings[10] fournit une plate-forme complète pour la construction d'applications indépendamment de l'intergiciel utilisé. Il s'appuie sur un système de composants orientés services qui utilisent des conteneurs pour l'exécution des services. Cependant OpenWings n'offre pas non plus de connecteurs dédiés au traitement de flots de mesures que nous avons identifié comme un besoin essentiel pour les services basés capteurs.

## 5. Perspectives et conclusions

Cet article a présenté SensorBean, un modèle à composants dédié au développement des services basés capteurs. Par rapport à d'autres modèles à composants liés à des plates-formes (EJB, .NET) ou bien universels (Fractal, Excalibur), SensorBean est centré sur le traitement des flots de mesures acquises par une myriade de capteurs dans une architecture distribuée multi plates-formes dynamique. Dans le modèle, l'unité d'assemblage est le composant. L'assemblage se fait au moyen de trois types de connecteurs : facette-réceptacle, sources et puits d'événements, et producteur-consommateur de données. Alors que les deux premiers types se retrouvent dans les principaux modèles à composants, le connecteur producteur-consommateur est centré sur l'échange des mesures acquises par les capteurs et les deux rôles peuvent avoir l'initiative de l'échange des mesures. Les connecteurs du modèle sont exprimés en terme de contrat (de service) qualifié pour assurer la propriété de substituabilité de composants conformément à l'approche orientée service. L'architecture des services est dynamique pour prendre en compte l'apparition et la disparition définitive ou temporaire des capteurs physiques ou des passerelles dans l'environnement.

L'expérimentation autour de SensorBean a démarré par la réalisation d'un démonstrateur s'exécutant sur des passerelles OSGi et des serveurs J2EE qui a été le point de démarrage de la réflexion pour la définition d'un nouveau modèle à composants. Dans le prototype, les connecteurs (inter plate-formes) entre les composants sont réalisés au moyen de Web Services. Nous avons développé un premier prototype de générateur de conteneur pour ce modèle pour la partie OSGi de l'architecture. Ce conteneur combine à la fois Fractal et des services OSGi développés dans l'équipe tels que ServiceBinder et WireAdminBinder. Le démonstrateur est disponible à l'adresse <http://www-adele.imag.fr/sensorbean/> et devrait être une des premières contributions au futur projet open-source d'intergiciels pour les Services Basés Capteurs ([sensor.objectweb.org](http://sensor.objectweb.org)). Ce modèle à composant SensorBean est maintenant une des pierres angulaires du projet RNRT PISE (Passerelle Internet Sécurisée et flexible).

## 6. Références

1. Szyperki (C.) – Component software : beyond object-oriented programming – ACM Press/Addison-Wesley Publishing Co., 1998.
2. PECOS : Pervasive Component Systems – <http://www.pecos-project.org/> – (IST-1999-20398).
3. ROBOCOP Project – <http://www.extra.research.philips.com/euprojects/robocop>.
4. Open Services Gateway Initiative – OSGi Service Platform (3d Release) – March 2003.
5. Plasil (F.) et al. – SOFA/DCUP : Architecture for Component Trading and Dynamic Updating – Proceedings of ICCDS 98, May 4-6, 1998, Annapolis, Maryland, USA. IEEE CS Press. 1998.
6. Cervantes (H.) et Hall (R.) – Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model – International Conference on Software Engineering (ICSE), May 2004, Edinburgh, Scotland.
7. Bieber (G.) et Carpenter (J.) – Introduction to Service-Oriented Programming (Rev 2.1) – Online Document, April 2001, (<http://www.openwings.org/download.html>).
8. Bruneton (E.) et al. – An Open Component Model and Its Support in Java – Proceedings of 7th International Symposium on Component-Based Software Engineering CBSE7, Edinburgh, Ecosse, 24-25 mai 2004, LNCS, vol. 3054, Springer-Verlag, 2004, p. 7-22.
9. Cervantes (H.) et al – FROGi : Déploiement de composants Fractal sur OSGi – 2004, ISBN 2-7261-1276-5.
10. OpenWings project – <http://www.openwings.org/download.html>