Self-controlled components for human-machine interaction services

Frédéric Lemoine
CEDRIC/CNAM
292 rue Saint-Martin, 75003 Paris, France
frederic.lemoine@cnam.fr

Tatiana Aubonnet
CEDRIC/CNAM
292 rue Saint-Martin, 75003 Paris, France
tatiana.aubonnet@cnam.fr

Noëmie Simoni
Télécom ParisTech
46 rue Barrault, 75013 Paris, France
simoni@telecom-paristech.fr

ABSTRACT

Cloud computing and Future Internet promise a new ecosystem where everything is "as a service", reachable and connectable anywhere and anytime, everyone succeeding in getting a service composition that meets his needs. Applications are now being constructed as micro-services compositions integrating more and more functionalities including human-machine interaction. We must answer the following questions: Can a human-machine interaction be implemented using microservices and can we control its behavior? Based on a service composition entity called Self-Controlled Component (SCC), we show how a human-machine interaction can be decomposed into elementary microservices. In addition, the self-controlling mechanisms integrated in the SCCs

IHM'17, August 29-September 1, 2017, Poitiers, France

ACM ISBN 978-1-4503-5109-6/17/08. https://doi.org/10.1145/3132129.3132153

make it possible to check the quality of service rendered for this interaction to introduce automation necessary for the dynamic reactions. The objective of this new concept is to provide a quality of service (QoS) guaranteed for the whole of a composite application.

CCS CONCEPTS

• **Human-centered computing** → *User interface management systems* • **Applied computing** → *Service-oriented architectures*

KEYWORDS

Human-Machine Interaction Services, Self-control, Quality of service, Service composition

RÉSUMÉ

Le cloud computing et l'internet du futur promettent un nouvel écosystème où tout serait "as a service", accessible et connectable partout et à tout moment. Chacun pouvant obtenir une composition de services répondant à ses besoins. Les applications sont maintenant construites comme des compositions de micro-services intégrants de plus en plus de fonctionnalités y compris celle de l'interaction homme-machine. Nous devons répondre aux questions suivantes : Une interaction homme-machine peut-elle être implémentée en utilisant des microservices et peut-on contrôler son comportement ? En nous basant sur une entité de composition de service appelée Self-Controlled Component (SCC), nous montrons comment une interaction homme-machine peut être décomposée en microservices élémentaires. De plus, les mécanismes d'autocontrôle intégrés aux SCCs permettent de vérifier la qualité de service rendue pour cette interaction afin d'introduire les automatismes nécessaires aux réactions dynamiques. L'objectif de ce nouveau concept est de fournir une qualité de service (QoS) garantie pour l'ensemble d'une application composée.

MOTS-CLEFS

Services d'Interaction Homme-Machine, Autocontrôle, Qualité de service, Composition de service

ACM Reference format:

Frédéric Lemoine, Tatiana Aubonnet and Noëmie Simoni. 2017. Composants autocontrôlés pour les services d'interaction homme-machine. In *Proceedings of 29eme Conference Francophone sur l'Interaction Homme-Machine , Poitiers- Futuroscope, FR, Août 2017 (IHM 2017),* 8 pages.
DOI: 10.1145/3132129.3132153

L'approche "mashup" consiste à combiner les données et les fonctionnalités. Un "mashup" est construit par l'assemblage et la combinaison de plusieurs fonctions existantes intégrée dans une nouvelle application [21]. Chaque utilisateur peut composer son propre service avec d'autres services afin d'en créer un nouveau. Cette architecture se compose de trois éléments [22] : données, Services et interface utilisateur (présentation). Ce concept a été popularisé par Google Maps (Maps.google.com) et Flickr (www.flickr.com). En publiant leurs API au public, ces derniers ont permis aux développeurs de les réutiliser pour créer de nouvelles applications. En restreignant le développement à la composition de fonctionnalités, les mahups ont permis de créer rapidement des applications personnalisées [18, 19]. Le mashup pour le Web [4, 14] compose des morceaux d'interfaces (parties de pages web ou widgets javascript) pour en créer de nouvelles.

1 INTRODUCTION

Le cloud computing et l'internet du futur promettent un nouvel écosystème où tout serait "as a service", accessible et connectable partout et à tout moment. Chacun pouvant obtenir une composition de services répondant à ses besoins. Les architectes migrent vers l'architecture orientée service (en anglais : Service Oriented Architecture, SOA). Nous sommes dans l'ère des services et le micro-service est au cœur de l'architecture. Les applications sont maintenant construites comme des compositions de micro-services [6, 5, 8] intégrant de plus en plus de fonctionnalités y compris celle de l'interaction homme-machine. Notre positionnement consiste à répondre aux questions suivantes :

- Peut-on concevoir une interaction homme-machine à l'aide de micro-services ?
- Comment vérifier la qualité de service rendue pour cette interaction ?
- Comment fournir une qualité de service garantie pour l'ensemble d'une application composée ?

Après avoir étudié les travaux connexes (Section 2), nous présentons dans la section 3 notre entité de composition de service appelée Self-Controlled Component (SCC). Nous montrons dans la section 4 comment concevoir et contrôler une interaction homme-machine en se basant sur ce composant. Un cas d'étude illustre nos propositions (Section 5). Finalement, une conclusion (Section 6) termine l'article.

2 TRAVAUX CONNEXES

Un système interactif est constitué d'un noyau fonctionnel (NF) et d'une Interface Homme-Machine (IHM). Le NF regroupe l'ensemble des traitements indépendamment de toute représentation à l'utilisateur. L'IHM fait les choix de présentation compte tenu du contexte d'usage courant et des propriétés ergonomiques à satisfaire. Les architectures telles que Arch [13], MVC [23] ou PAC [16] séparent le NF et l'IHM. Plusieurs approches sont apparues afin de composer une Interface Homme-Machine comme l'approche "mashup", les portlets [1], les infrastructures d'interface utilisateur Windows Presentation Foundation (WPF) [9] ou les Java Server Faces (JSF) [7].

La composition du noyau fonctionnel (NF) est étudiée en génie logiciel, et plus précisément dans les approches à composants et à services. Plusieurs approches ont été proposées : Approche à objets [17], à composants [25, 15] et orienté services [2]. L'approche orientée service vise à apporter une grande flexibilité au développement des applications [2] et permet d'éliminer les dépendances entre les différents éléments d'une application. L'idée générale est de créer des applications modulaires à liens lâches permettant d'adapter le développement des applications aux besoins. S'appuyant sur une architecture SOA (Service Oriented Architecture), le service représente l'unité d'abstraction pour le développement des applications.

Les différentes approches citées proposent de composer, de juxtaposer des composants de manière horizontale à la manière d'un puzzle afin notamment de les assembler pour concevoir une

nouvelle interface. Aucune ne propose une gestion verticale de l'interface. C'est-à-dire de concevoir une interaction homme-machine complète sous forme d'une composition de services intégrant à la fois le rendu mais aussi la gestion des interactions.

De plus, même si toutes les approches étudiées traitent de la conception d'interfaces ou du noyau fonctionnel par composition, aucune approche n'intègre le contrôle du bon fonctionnement de son interaction avec l'utilisateur et n'offre une qualité de service (QoS) garantie.

3 TRAVAUX PRÉCÉDENTS

Dans le monde du service, le service est au cœur de l'architecture, pour profiter de tous les avantages attendus de ce concept, nous avons proposé dans [12] un composant appelé Self-Controlled service Component (SCC), dont nous rappelons ici la description.

Un composant encapsule un service unique. Ces services peuvent être de tailles et de natures différentes : service d'analyse d'image en temps réel (reconnaissance de visages, de caractères, code barre) dans le cas de l'internet des objets, algorithme (tri, cryptage), capture d'images, etc.

Pour décrire le comportement de nos composants et permettre une gestion homogène de la qualité de service (QoS), nous définissons un modèle de QoS générique [24]. Quatre critères sont proposés pour décrire la QoS: la disponibilité (taux d'accessibilité du composant de service), l'intégrité (capacité à s'exécuter sans altération de l'information), le temps (temps de traitement) et la capacité (charge maximale que le composant de service peut gérer). Cela s'est révélé utile et suffisant dans tous les cas pratiques que nous avons étudiés. Pour augmenter la décomposition structurelle et la réutilisation des composants de QoS non fonctionnels, nous avons séparé ses fonctions internes et proposé une architecture qui sépare les fonctions de surveillance et de QoS du reste des fonctions de base.

Nous avons spécifié ce modèle dans le projet OpenCloudware [3] pour aborder les aspects comportementaux au travers de la QoS (Figure 1).

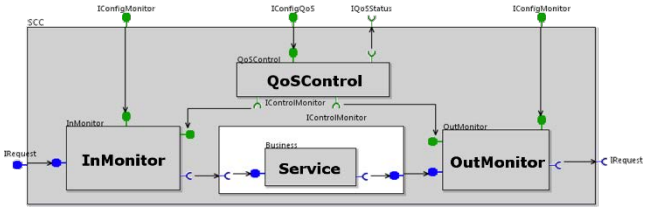

Figure 1: Self-Controlled service Component (SCC)

Les composants de surveillance de l'entrée (InMonitor) et de la sortie (OutMonitor) jouent un rôle d'intercepteur. Les requêtes de service entrantes sont interceptées et transmises (inchangées) au composant fonctionnel. Ils fournissent des informations de mesure sur le flux qu'ils

interceptent. Le composant QoSControl vérifie le comportement courant de la ressource et sa conformité au contrat. Pour cela, il demande régulièrement aux moniteurs (InMonitor et OutMonitor) les valeurs de paramètre (méthode getValues). Il compare chaque valeur courante à la valeur de seuil correspondante à ne pas dépasser. Il envoie une notification OutContract si la valeur courante est inférieure (ou supérieure) à la valeur de seuil. Dans ce cas, la gestion dynamique consiste à remplacer à la volée le composant défaillant par un service ubiquitaire répondant aux exigences. Sinon, il envoie une notification InContract. Nous obtenons un composant SCC, auto-monitoré et autocontrôlé. Les sous-composantes de la membrane (moniteurs et QoS) sont activés pour effectuer le monitoring de la qualité de service et notifier son bon/mauvais fonctionnement. Ce composant a été conçu de manière à pouvoir être composé avec d'autres pour former une composition.

4 PROPOSITIONS POUR LA QUALITE DE SERVICE D'UNE INTERACTION HOMME-MACHINE AUTOCONTROLEE

En nous basant sur notre SCC, nous allons montrer dans ce chapitre comment :

- Concevoir une interaction homme-machine à l'aide de micro-services.
- Vérifier la qualité de service rendue pour cette interaction.
- Fournir une qualité de service (QoS) garantie pour l'ensemble d'une application composée (interface et noyau fonctionnel).

La qualité de service que nous avons définie dans nos travaux précédents reste ici valable pour les interactions homme-machine. Nous gardons donc la même définition (disponibilité, intégrité, temps et capacité). Dans le cloud computing, les plateformes de services et l'Internet des objets (IoT) le composant est la pierre angulaire. Chaque application (composition de service) répond à une demande du client (réactivité, disponibilité...) basée sur des ressources et sur ce que peux lui apporter son environnement. Nous avons montré comment créer une application à l'aide de composants de services autocontrôlés [11]. Nous allons maintenant montrer que notre approche basée sur des composants SCC (Figure 1) peut être étendue à la conception d'interfaces interactives.

Une interface homme-machine se compose d'un dispositif d'acquisition (boutons, souris, gants, molettes, stylet pour la reconnaissance d'écriture, joysticks, clavier, surface tactile, télécommande, microphone pour les commandes vocales, capteurs de mouvement, etc.) et d'un dispositif de restitution (écrans, lampes témoins/voyants d'état, retour d'effort, haut-parleur, etc.). Une interface tangible où l'utilisateur interagit avec l'information numérique par le moyen de l'environnement physique, peut être composée par exemple d'une surface tactile (acquisition) couplée à un écran (restitution). Un dispositif d'acquisition est composé de plusieurs capteurs. Une méthode permettant de transformer un dispositif intelligent en un composant as-a-service comme le SCC a été proposée dans [10]. L'interface de restitution doit être indépendante du reste de l'application dans le sens où elle est interchangeable avec d'autres. Nous n'aurons pas, en effet, la même interface si nous utilisons un téléphone portable dont la surface d'affichage est limitée ou si

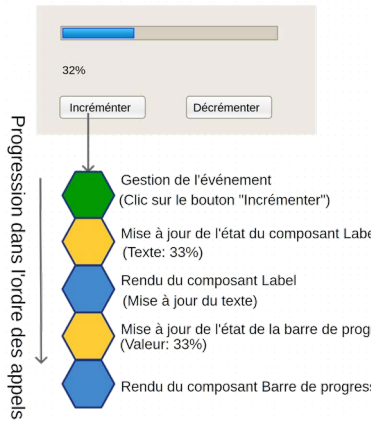

Figure 2 : Gestion d'un clic à l'aide d'une composition de services

nous utilisons un ordinateur de bureau ayant une surface bien plus grande même si la quantité d'information à afficher est la même. Le micro-service responsable de la restitution sera donc différent suivant l'appareil utilisé par l'utilisateur. Une application complète peut être conçue en composant des micro-services (Figure 2).

Une interface homme machine comporte au moins trois types de micro-services :

- Gestion des évènements du dispositif d'acquisition
- Changement d'état
- Rendu d'interface.

Le *micro-service de gestion des évènements du dispositif d'acquisition (en vert)* traite l'interaction avec l'utilisateur. Il prend en compte les événements liés aux différents capteurs utilisés. Exemple : On clique sur un élément placé sur une surface tactile, un nouveau caractère a été entré au clavier, le bouton de la souris a été enfoncé, un son a été capté par le microphone, etc. Le *micro-service de changement d'état (en jaune)* permet de changer l'état d'un élément du dispositif de restitution. Exemple : l'élément sélectionné dans une liste ou le pourcentage atteint dans une barre de progression dans le cas d'écran graphique, l'allumage ou non d'une lampe témoin, le son à émettre par le haut-parleur, etc. Le *micro-service de rendu d'interface (en bleu)* est chargé de la restitution. C'est-à-dire du dessin du composant dans le cas d'interface graphique, de l'allumage de la lampe témoin, de l'émission du son par le haut-parleur, de la production d'un retour d'effort, etc. Chacun de ces trois micro-services sera un SCC.

La Figure 2 montre une interface graphique classique composée de quatre éléments. Deux boutons "Incrémenter" et "Décrémenter" incrémentent/décrémentent le pourcentage d'une barre de progression ainsi que sa valeur affichée au format texte. L'acquisition se fait à l'aide d'une souris. À chaque élément, nous pouvons associer une composition de micro-services responsables du dessin, du changement de son état et de la gestion des évènements. Chaque micro-service étant un SCC, nous contrôlons son service rendu (QoS). De même, la composition complète étant SCC (Moniteurs d'entrées/sorties et QoSControl), nous contrôlons également la qualité de service rendu par celle-ci. Nous mesurons en particulier le temps de traitement suite au clic sur le bouton et donc la réactivité de l'interface vue de l'utilisateur. Nous savons détecter directement quel service est défaillant puisque, celui ne remplissant pas son contrat (temps de traitement supérieur à la valeur de seuil par exemple), enverra un OutContract. La composition complète est ainsi elle-même contrôlée.

Comme nous l'avons dit plus haut, le micro-service de rendu est interchangeable et dépend du dispositif utilisé par l'utilisateur. La composition s'adapte à l'utilisateur. Les services de la composition ne sont pas nécessairement situés en totalité sur le dispositif de l'utilisateur mais peuvent se situer ailleurs (Gateway, cloud...) [10]. De même, le micro-service proposé peut-être local. Le service prenant part à la composition peut, en effet, être celui le plus proche géographiquement de l'utilisateur. Celui-ci établit une session (composition) à partir d'un ensemble de micro-services dont certains sont proches de lui améliorant ainsi le délai de communication ou offrant un service propre à sa situation géographique (Horaires des trains,

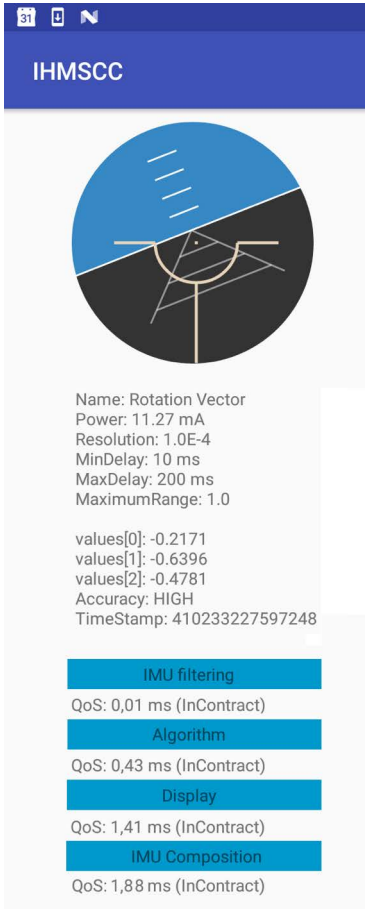

Figure 3 : Implémentation de l'horizon artificiel à l'aide de SCCs.

avions, météo, etc.). Notez que la procédure à appliquer après la détection d'un OutContract est hors de portée de ce document cependant plusieurs directions à explorer ont été exposées dans [20].

5 CAS D'ETUDE

Nous allons montrer l'intérêt de notre approche et sa faisabilité sur un cas d'étude. Nous allons donc réaliser une maquette qui servira de preuve de concept représentant un système de navigation aéronautique équipé de nombreux capteurs (gyroscopes, accéléromètres, magnétomètres, etc.). L'idée est d'afficher une représentation des données issues de ces capteurs et de montrer que nous contrôlons la chaine complète de l'affichage et de l'interaction.

5.1 Matériel utilisé pour la mise en œuvre

Nous avons choisi de porter notre approche sur un type de matériel hétérogène comportant des capteurs comme tout système critique permettant de rendre compte de la réalité d'une situation (altitude, orientation, etc.). Nous avons choisi pour cela une tablette Nexus 9 fonctionnant sur Google Android. Cette plateforme intègre une centrale à inertie IMU (Inertial Measurement Unit). En utilisant une combinaison d'accéléromètres, de gyroscopes et parfois aussi des magnétomètres, une centrale à inertie ou centrale inertielle est un instrument utilisé en navigation pour estimer l'orientation d'un mobile (angles de roulis, de tangage et de cap), sa vitesse linéaire et sa position. Les centrales à inertie sont habituellement utilisées pour manœuvrer des avions, des véhicules aériens non habités (UAV), des engins spatiaux y compris les satellites et les atterrisseurs.

5.2 Conception de l'interface

Nous allons donc réaliser une maquette, une illustration sur un cas concret, représentant un système de navigation aéronautique simplifié. Il ne comprendra qu'un horizon artificiel. Chaque composant d'interface est défini comme une composition de services autocontrôlés (SCC). Ainsi l'horizon artificiel est composé de trois SCCs (Figure 4) permettant : (i) La réception des mesures (ce SCC est chargé de réceptionner les mesures effectuées par la centrale à inertie et de les filtrer en éliminant les mesures n'ayant pas un niveau de précision suffisant), (ii) Le calcul du nouvel état du composant (prépare la représentation graphique du composant à l'aide des données du premier SCC) et (iii) Le rendu du composant (dessine l'horizon artificiel à l'aide des données du SCC précédent). Chaque SCC est autocontrôlé. Nous contrôlons chaque service individuellement et savons s'il respecte son contrat. De même nous contrôlons la composition complète formée de ces 3 SCC. Ainsi nous garantissons que l'affichage vu par le pilote correspond bien à la réalité mesurée par les capteurs. Le délai entre les mesures et leur représentation sur l'écran est en effet limité et contrôlé. La Figure 3 montre l'implémentation de l'interface sur la tablette. Nous mesurons le temps de traitement (QoS) de chaque SCC ainsi que celui de la composition. Ces valeurs sont

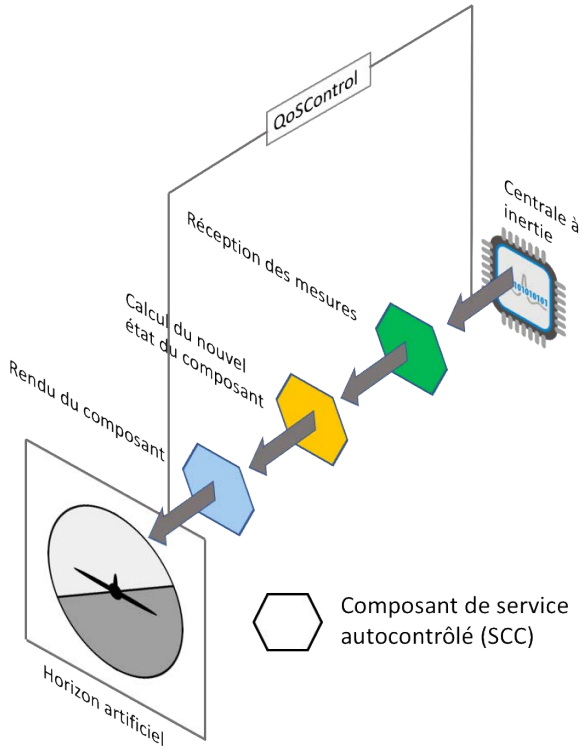

Figure 4 : Réalisation de l'interface de l'horizon artificiel par une composition de services autocontrôlés (SCC)

6 CONCLUSION

Nous avons montré comment une interaction homme-machine peut être décomposée en micro-services élémentaires en nous basant sur l'entité de composition de service appelée Self-Controlled Component. Nous avons également montré comment les mécanismes d'autocontrôle intégrés aux SCCs permettent de vérifier la qualité de service rendue pour cette interaction. Ce nouveau concept permettra ainsi de fournir une qualité de service garantie pour l'ensemble d'une application composée de micro-services.

RÉFÉRENCES

[1] 2003. JSR 168: Portlet Specification. (2003). https://www.jcp.org/en/jsr/detail?id=168

[2] 2007. Service Component Architecture (SCA) | OASIS Open CSA. (2007). http://www.oasis-opencsa.org/sca

[3] 2015. The OpenCloudware project. (2015). http://www.opencloudware.org/

[4] 2016. Web Services for Remote Portlets. (2016). https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp

[5] 2017a. Amazon Web Services. (2017). https://aws.amazon.com/fr

[6] 2017b. IBM Bluemix. (2017). https://www.ibm.com/cloud-computing/bluemix/fr

[7] 2017c. JavaServer Faces Technology. (2017). http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[8] 2017d. Microsoft Azure. (2017). https://azure.microsoft.com/fr-fr/

[9] 2017e. Windows Presentation Foundation. (2017). https://docs.microsoft.com/en-us/dotnet/framework/wpf/

[10] Tatiana Aubonnet, Amina Boubendir, Frédéric Lemoine, and Nöemie Simoni. 2016. Controlled Components for Internet of Things As-A-Service. Open Journal of Internet Of Things (OJIOT) 2, 1 (2016), 16–33. https://www.ronpub.com/OJIOT-2016v2i1n02_Aubonnet.pdf

[11] Tatiana Aubonnet, Ludovic Henrio, Soumia Kessal, Oleksandra Kulankhina, Frédéric Lemoine, Eric Madelaine, Cristian Ruz, and Noëmie Simoni. 2015. Management of service composition based on self-controlled components. Journal of Internet Services and Applications 6, 15 (2015), 17. http://dx.doi.org/10.1186/s13174-015-0031-7

[12] Tatiana Aubonnet and Noëmie Simoni. 2014. Self-Control Cloud Services. In 2014 IEEE 13th International Symposium on Network Computing and Applications, NCA 2014, Cambridge, MA, USA, 21-23 August, 2014. 282–286. http://dx.doi.org/10.1109/NCA.2014.48

[13] L Bass and Joelle Coutaz. 1992. A metamodel for the runtime architecture of an interactive system: the uims tool developers workshop. In SIGCHI Bulletin. Number 24. 32–37.

[14] Wolfgang Beinhauer and Thomas Schlegel. 2005. User Interface for Service Oriented Architectures. In Intelligent Production Machines and Systems.

[15] Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, Vivien Quéma, and Jean-Bernard Stefani. 2006. The FRACTAL component model and its support in Java. Software: Practice and Experience 36, 11-12 (Sept. 2006), 1257–1284. 1097-024X http://dx.doi.org/10.1002/spe.767

[16] Joelle Coutaz. 1987. PAC, an Object Oriented Model for Dialog Design. In Interact'87. Stuttgart.

[17] Booch Grady. 1994. Object-Oriented Analysis and Design With Applications. (1994).

[18] Lars Grammel and Margaret-Anne Storey. 2008. An end user perspective on mashup makers. University of Victoria Technical Report DCS-324-IR (2008). https://sites.google.com/site/larsgrammel/paper_mashup_makers.pdf

[19] Björn Hartmann, Scott Doorley, and Scott R. Klemmer. 2008. Hacking, mashing, gluing: Understanding opportunistic design. IEEE Pervasive Computing 7, 3 (2008). http://ieeexplore.ieee.org/abstract/document/4563909/

[20] Frederic Lemoine, Tatiana Aubonnet, Ludovic Henrio, Soumia Kessal, Eric Madelaine, and Noemie Simoni. 2017. Monitoring as-a-service to drive more efficient future system design. EAI Endorsed Transactions on Cloud Systems 17, 9 (6 2017). http://dx.doi.org/10.4108/eai.28-6-2017.152754

[21] D. Lizcano, J. Soriano, M. Reyes, and J. J. Hierro. 2008. EzWeb/FAST: Reporting on a Successful Mashup-Based Solution for Developing and Deploying Composite Applications in the Upcoming #147;Ubiquitous SOA #148;. In 2008 The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. 488–495. http://dx.doi.org/10.1109/UBICOMM.2008.61

[22] Duane Merrill. 2006. Mashups: The new breed of Web app. IBM Web Architecture Technical Library (2006), 1–13.

[23] Trygve Reenskaug. 1978. Mvc xerox parc. (1978). http://heim.ifi.uio.no/trygver/themes/mvc/mvc-index.html

[24] Tatiana Aubonnet and Noëmie Simoni. 2013. Service Creation and Self-management Mechanisms for Mobile Cloud Computing. In Wired/Wireless Internet Communication - 11th International Conference, WWIC 2013, St. Petersburg, Russia. Proceedings. 43–55. http://dx.doi.org/10.1007/978-3-642-38401-1_4

[25] Jorge Villalobos. 2003. Fédération de composants: une architecture logicielle pour la composition par coordination. Ph.D. Dissertation. Université Joseph-Fourier-Grenoble I. https://tel.archives-ouvertes.fr/tel-00004378/