Business Process Personalization through Web Widgets

Nassim Laga, Emmanuel Bertin
Orange Labs
France Telecom R&D, 42, rue des Coutures,
14000 Caen France
{nassim.laga, emmanuel.bertin}@orange-ftgroup.com

Noel Crespi
Institut Telecom, Telecom SudParis,
9 rue Charles Fourier, 91011, Evry Cedex, France
noel.crespi@it-sudparis.eu

Abstract— **Widget aggregators such as iGoogle and Netvibes are broadly adopted by the mass market. They enable end-users to personalize their environment with their preferred services (Widgets). However, the usage in an enterprise context is not yet investigated. In this paper, we firstly show that in addition to personalization capability, the integration of business processes should be considered. Secondly, we propose a new Widget aggregator that enables the end-user to personalize a business process by chaining Widgets according to his/her needs and habits. Thirdly, we introduce a new approach for specifying an end-user process; an approach which enables even ordinary end-users, without computing skills, to define their processes. Finally, we validate these concepts by implementing and testing a prototype. As a consequence, this work does not only impact Widget aggregators, but it also innovates in end-user service creation research by proposing an intuitive tool, understandable even by ordinary end-users, for specifying their processes (composite services).**

Keywords-Personalization; Business processes; Web 2.0; Mashups; SOA

I. INTRODUCTION

Widget aggregators like iGoolge [1] and Netvibes [2] are popular mass market Web applications. They enable end-users to create a single and personalized web page to access several services. However, their usage within an enterprise context is not yet investigated. In [3] we have described and implemented a Widget aggregator. It has been experimented among 184 participants within Orange Labs, and 63% of end-users have reported the need for integrating their enterprise applications and processes to the Widget aggregator.

A business process is "*the combination of a set of activities within an enterprise with a structure describing their logical order and dependence whose objective is to produce a desired result*" [4]. These business processes are usually specified using graphical tools such as BPMN [5], and implemented by developers using for example BPEL4WS [6 and 7]. Thus, end-users are not involved within the procedure of specifying and integrating a business process.

However, business processes might also depend on end-user personal needs [8 and 9]. Consider for instance a vacation request business process. An end-user may want to automatically update his/her agenda for each confirmed vacation request, while another one may want to automatically send an email to his/her collaborators to notify them about his/her unavailability. To tackle this heterogeneity of business processes, current approaches are developer-centric; they aim to accelerate the application creation process in order to respond quickly to the end-users needs. Current Service-Oriented Architecture (SOA) [10] for instance enables developers to implement quickly new services using other ready-to-use services developed by third party entities. We witness even the emergence of composition languages, such as BPEL4WS [6] and SPATEL [11], that enable graphical-based composition, and thus facilitate and significantly speed up service creation processes. However, though this is a successful approach for implementing long-lived processes, it is not adapted for the implementation and integration of dynamic and end-user-dependent processes. Therefore, we propose in this paper to enable directly end-users to personalize business processes according to their needs. This is in line with current trends [8 and 12] that consider knowledge workers as co-producers of software features. As a consequence, the task of defining and implementing a business process is scattered over the developers and the end-users. Developers are in charge of specifying and implementing the generic and long-lived part of a business process (the part which is common to a significant population of end-users), and the end-user is responsible for the part of the business process which is specific to him/her.

Our approach relies on the Widget aggregation paradigm. The framework we define and implement is a Widget aggregator that enables Widgets to communicate with each other in order to allow the end-user to fulfill a process that he/she previously designed. The specificity of our approach is twofold: firstly, we introduce an innovative approach for specifying end-user personal processes – an approach which is intuitive and understandable even by ordinary end-users, and secondly, we enable independent Widgets to exchange information according to these processes. Furthermore, by relying on Widget paradigm, we also promote human-to-machine interaction, in addition to machine-to-machine interaction facilities provided by SOA.

In section (2) of this paper we illustrate the motivations for enabling end-users to configure their Widget aggregator according to their business processes. Then, we summarize the use cases of the framework we propose. In section (4) we detail its design and implementation. We validate the solution in section (5). We discuss the positioning of our work regarding service creation technologies and Widget aggregators in section (6). Finally, we conclude the paper in section (7).

II. Incentive Example

In this section we figure out, through concrete use case, the benefits that come from using Widgets and Widget aggregators as the basis for implementing and integrating business processes. More precisely, we firstly illustrate that business processes contain two parts: a common part and an end-user specific part. And secondly, we introduce our approach which is characterized by enabling end-users to manage themselves their specific part of the processes.

Let's consider an end-user who has personalized his/her Widget aggregator by loading some services such as: sending email, reading and updating the agenda, telephony, and a vacation request management service. The vacation request management service is a business process that involves one or several business entities. Current technologies such as BPMN, SOA, and BPEL4WS enable modeling and implementing such business processes. But, they do not enable the end-user to personalize it according to his/her specific needs. Indeed, after receiving a response for a vacation request, the end-user might need to launch other activities which involve other services such as: updating his/her agenda by entering his/her unavailability during the leave period, setting up an automatic email response during the leave period, setting up an incoming calls redirection to the voice mail during the leave period and sending email to his/her collaborators to notify them about his/her unavailability. Such actions are end-user-dependent, which makes them almost impossible to automate by developers. Figure 1 for instance illustrates the actions that might be undertaken by two different profiles of end-users: a team manager and a purchasing and logistics responsible; actions which are completely different.

Figure 1 clearly illustrates that business processes actually comprise two parts: a part which is common to a significant population of end-users, and a part which is heterogeneous, dynamic, and specific to a limited number of end-users. The end-user specific part may depend not only on the role of the end-user but also on his/her personal habits. While the automation of the common part of a process is well addressed by current tools, the end-user specific part can not be generalized, and thus almost impossible to automate by the developers. As a consequence, we obviously need a more flexible mechanism that enables directly end-users to specify and automate the end-user specific parts of business processes.

Though business processes could be implemented and automated using a dedicated application, the current trends rely on service composition. They are currently specified using a dedicated environment such as Eclipse BPEL editor. However, while these tools are suited for developers and

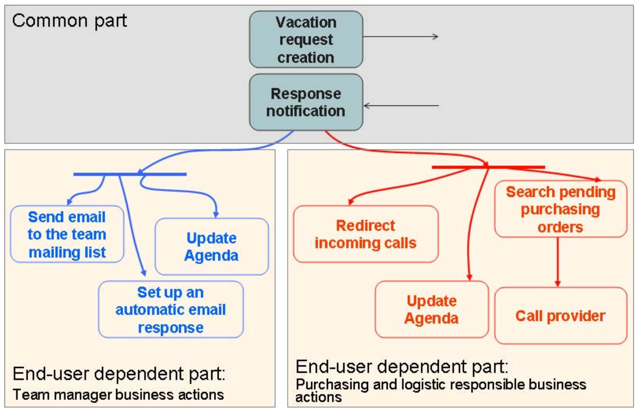

Common part
Vacation request creation
Response notification
Send email to the team mailing list
Update Agenda
Set up an automatic email response
End-user dependent part:
Team manager business actions
Redirect incoming calls
Search pending purchasing orders
Update Agenda
Call provider
End-user dependent part:
Purchasing and logistic responsible business actions

Fig. 1. Business actions heterogeneity illustration.

advanced end-users, we can not expect from ordinary end-users to master and use them. Consequently, we propose in this paper to enable the end-users to specify and execute their processes directly within their Widget aggregator environment, which is also their working environment. This is an enhancement to current Widget aggregators such as iGoogle [1] and Netvibes [2] which do not enable services to communicate with each other in order to perform an end-user specific process. Even if some frameworks such as [12, 13, and 14] include inter-widget communication tool, they are either not based on end-users processes [13 and 14], or the process specification tool is too complex to be used directly by ordinary end-users [12].

From the technical perspectives, our approach is characterized by firstly wrapping functionalities of services and applications within Widgets. Secondly, we enable the end-user to load only the functionalities (Widgets) he/she needs. Finally, we enable him/her to configure the working environment (the Widget aggregator) so that Widgets collaborate with each other in order to support him/her when performing his/her business goals; this configuration is essentially the specification of the end-user specific part of his/her business processes. The specificity of our approach is the definition and implementation of a new process definition method which is intuitive and understandable even by ordinary end-users. This tackles the heterogeneity as well as the dynamicity of the end-user specific part of business processes.

Our contribution might be considered as a second level of personalization of Widget aggregators. The first level is personalizing the working environment by accessing only the services that an end-user needs, and the second one is personalizing the communications between services according to end-user specific processes. Figure 2, illustrates the end-user view of a personalized working environment. It illustrates a vacation request business process personalized by a “Purchasing and logistic” responsible (see Figure 1). The displayed Widgets, and links between them, are those that are involved in the personalized process. The common part of the business process is performed using the “Vacation request” Widget.

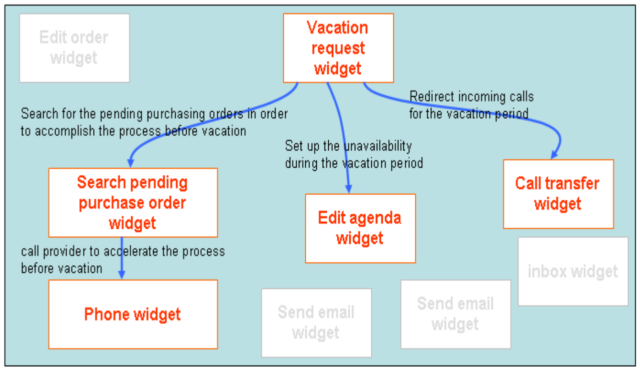

Edit order widget
Vacation request widget
Search for the pending purchasing orders in order to accomplish the process before vacation
Redirect incoming calls for the vacation period
Set up the unavailability during the vacation period
Search pending purchase order widget
Edit agenda widget
Call transfer widget
inbox widget
call provider to accelerate the process before vacation
Phone widget
Send email widget
Send email widget

Fig. 2. Example of purchasing and logistic responsible environment.

III. FRAMEWORK USE CASES

As we illustrate in Figure 3, the framework we propose has three use cases. The first use case, “Widget Deployment”, enables service providers to publish new Widgets into the platform. The second use case, “Business Process Definition”, includes two sub-use cases: the “Common Part Definition” use case and the “End-user Specific Part Definition” use case. The former enables business process management entities to specify the common part of a business process, which is generic and common to a significant population, and the latter enables the end-users to specify themselves the parts of business processes which are specific to them. Finally, the third use case, “Widget Aggregator Usage”, enables end-users to access and use their environment which is now enriched with the preferred Widgets which communicate with each other according to the specified business processes (common and end-user specific parts).

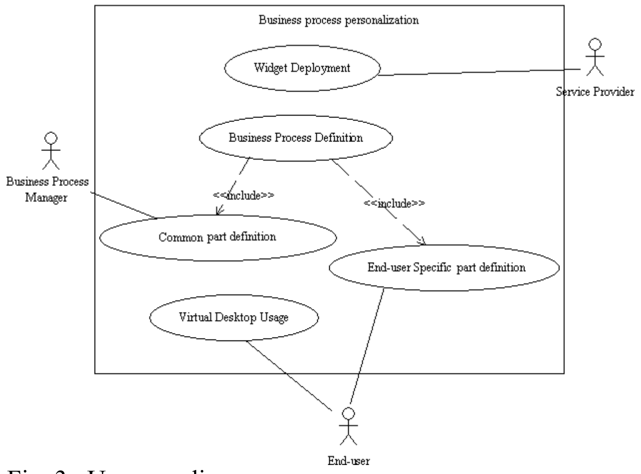


Fig. 3. Use case diagram.

Our contribution in this paper is mainly an end-user tool for defining and executing the end-user specific part of business processes. Consequently, in the following sections, we do not detail how the “Common Part Definition” use case is performed. We assume that current service composition tools such as BPEL4WS [6] and SPATEL [11] are tailored for this task, as we discuss in Section 6.

IV. FRAMEWORK DESIGN AND IMPLEMENTATION

In this section we detail the architecture and the implementation details of the framework we propose. Firstly, we define a Widget in an enterprise context. Secondly, we summarize the architecture and the implementation of the Widget aggregator. Thirdly, we detail the definition of the end-user specific part of a business process. Finally, we detail how processes are executed within the Widget aggregator.

A. Widget in Enterprise Context

Experimentations we made within Orange Labs have shown clearly the need for integrating corporate applications into the Widget aggregator we have implemented [3]. However, W3C definition of Widgets [15] is limited to a user interface (UI) that displays data. Therefore, we propose in this paper to enhance it and define a Widget as “a small client-side web application **for offering atomic functionalities of a service**, packaged in a way to allow a single download and installation on a client machine, mobile phone, or mobile Internet device”.

This new definition of a Widget enables software developers to expose functionalities of enterprise applications to the end-users. Thus, instead of exposing a single application that packages all functionalities, developers split their application into independent functionalities, and expose them as independent Widgets; each functionality is wrapped within a Widget. This is analogous to service oriented computing (SOC) [16] where applications are exposed as Web Services, except that in our proposal we expose Widgets (UI + functionality). Figure 4 shows some Widgets that have been implemented within Orange Labs.

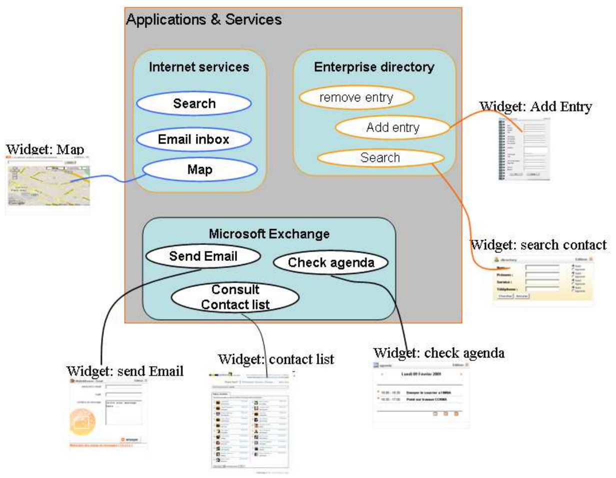


Fig. 4. Widgets paradigm.

Our proposal in this paper is to use Widget aggregation (detailed in Subsection 4.2) paradigm not only to personalize the end-user environment according to the Widgets he/she needs, but also to enable these Widgets to communicate with each other according to end-user specific processes (detailed in Subsection 4.3 and 4.4).

Technically, Widgets are defined using an XML file (provided when deploying a new Widget into the framework). It contains mainly the URL of the Widget, its inputs semantic tags, and its outputs semantic tags. The inputs and outputs semantics are defined using *Microformats* [1] [17 and 18] vocabulary; a lightweight semantic approach. We use for instance *hCard*[2] to represent contact information, and *hCalendar*[3] to represent calendar events. We rely on *Microformats* to semantically annotate, at the UI level, the outputs of each Widget. This enables us to chain the Widgets at the UI level.

B. *Widget Aggregator*

The Widget aggregator we propose is implemented as a Web application. It is mainly based on AJAX technologies (Asynchronous JavaScript And XML) [19]. We have used Dojo JavaScript library[4] to facilitate the integration of the Widgets into the Web page, and also to avoid cross-browser issues.

Figure 5 illustrates the different components that constitute the Widget aggregator. Some of them perform the aggregation of Widgets, while others are required for the process definition and execution use case, which are detailed in the next subsections.

Firstly, the Widget aggregator manages a database of users, Widgets, and processes. Using this database, the framework associates processes to end-users; processes which also define the useful Widgets to display within the end-user environment. This database is queried through "Persistence Management Component" which is implemented using PHP language.

Secondly, we have implemented the "End-user Specific Process Management Component" (ESPMC), which is in charge of creating, saving, and executing end-user specific part of processes. It interacts with the database to save and retrieve process definitions, and with "Widget Management Component" (WMC) to define and execute processes. A detailed description of ESPMC is provided in the next subsections.

Finally, we have implemented a "Widget Management Component" (WMC), which is in charge of displaying a Widget within the end-user environment. It is a Dojo object which is created from a Widget definition. WMC uses AJAX requests to load the UI of the Widget, and to interact with its server side logic.

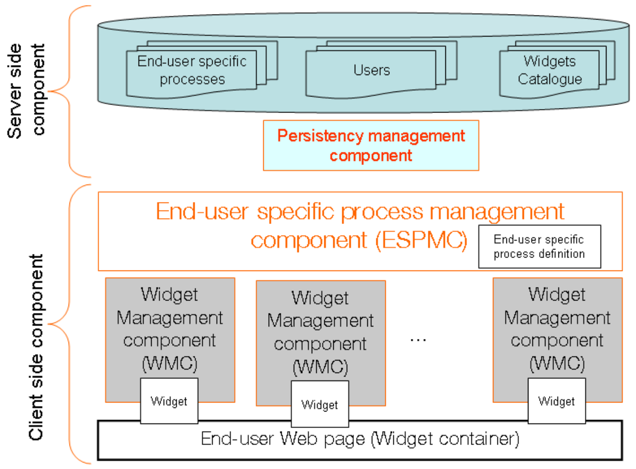

Server side component
End-user specific processes
Users
Widgets Catalogue
Persistency management component
Client side component
End-user specific process management component (ESPMC)
End-user specific process definition
Widget Management component (WMC)
Widget
Widget Management component (WMC)
Widget
...
Widget Management component (WMC)
Widget
End-user Web page (Widget container)

Fig. 5. Component view of the framework.

C. *Definition of the End-user Specific Part of a Business Process*

In this section, we firstly detail the language we use for defining processes, and then we describe the innovative approach for enabling ordinary end-users to specify their own business process.

1) Process definition: The end-user specific part of a business process is defined as a set of Widgets and links between them. Each link is defined by the source Widget, the destination Widget, the type of the link (automatic or semi-automatic), and the data and/or event which should be transmitted from the source Widget to the destination Widget.

There are two types of links: automatic links and semi-automatic links. Automatic links are executed without any initiative from the end-user. Each time the data and/or event that should be transmitted from the source Widget to the destination Widget are detected, the destination Widget is automatically launched without any direct initiative from the end-user. Semi-automatic links however are firstly displayed within the UI of the source Widget using HTML elements (typically an icon), and secondly the corresponding data are transmitted (from the source Widget to the destination Widget) only when the end-user clicks on that HTML element.

From the technical perspective, end-user specific processes are defined using a JSON format (RFC 4627 [20]). It facilitates and speeds up the creation and the interpretation of processes in the Web Browser. Table I details the definition of processes using JSON.

[1] Microformats, http://microformats.org/, accessed Feb 27th, 2010
[2] Microformats, http://microformats.org/wiki/hcard, accessed Feb 27th, 2010
[3] Microformats, http://microformats.org/wiki/hcalendar, accessed Feb 27th, 2010
[4] Dojo toolkit, http://www.dojotoolkit.org/, accessed Feb 27th, 2010

TABLE I JSON Definition Of Processes

	JSON format
Widgets	[{ widgetId: *value*, widgetName: *value*, widgetUrl: *value*, inputs: [{ inputSemanticTag: *value*, }, …], outputs: [{ outputSemanticTag: *value*, }, …], }, …]
Links	[{ linkId: *value*, sourceWidgetId: *value,* destinationWidgetId: *value,* linkType: *value*, sourceOutput: *value*, destinationInput: *value*, }, …]

2) Innovative approach for an end-user process definition: As we have mentioned in the introduction, we define and implement in this paper an innovative approach for specifying end-user specific part of business processes. It is innovative because it enables even ordinary end-users, without programming skills, to specify processes. This approach is characterized by firstly creating a "mesh" process as the end-user loads Widgets into his/her environment; a "mesh" process is a process that connects all connectable Widgets. This is possible and scalable because the "mesh" process is created only between Widgets that are loaded by the end-user. Secondly, starting from this "mesh" process, the end-user can delete undesired links or modify their type (automatic or semi-automatic). Finally, the end-user can save the process and optionally share it with other end-users. Figure 6 shows the interactions between different components of the aggregator to enable the end-user to create and save a process.

From the technical perspective, each time the end-user loads a new Widget into his/her environment, an event is sent to "End-user Specific Process Management Component" (ESPMC), which firstly detects semantic matching between the new Widget and other Widgets that are already loaded, and secondly creates automatically the corresponding links (Step 1 and 2). The detection of semantic matching between two Widgets is based on the *Microformats* tags used to annotate the inputs and the outputs of Widgets. For instance, if one Widget (e.g. directory) generates an *hCard*, which contains contact information (e.g. phone number), and another declares that it can receive as input a phone numbers (annotated by "*tel*"), a link will be automatically created between the two Widgets. At the execution, ESPMC will automatically extract the phone number from the *hCard* and launches the destination Widget with the phone number as an input parameter.

New links are by default semi-automatic. Then, the end-user can modify the type of a link or delete it (Step 4). Each time such action is performed, the ESPMC is informed, and the process definition is updated (Step 5 and 6). Finally, the end can save and share a created process (Step 7, 8, and 9). A common database is used for this purpose (saving and sharing a process definition).

This approach is intuitive because end-users do not have to think about creating a new link. Instead, when end-users realize that a link between two Widgets is not needed, they can delete it or modify its type. Moreover, if a given functionality is needed and not considered yet in the process, the end-user has just to load it; links between this functionality and others are automatically created.

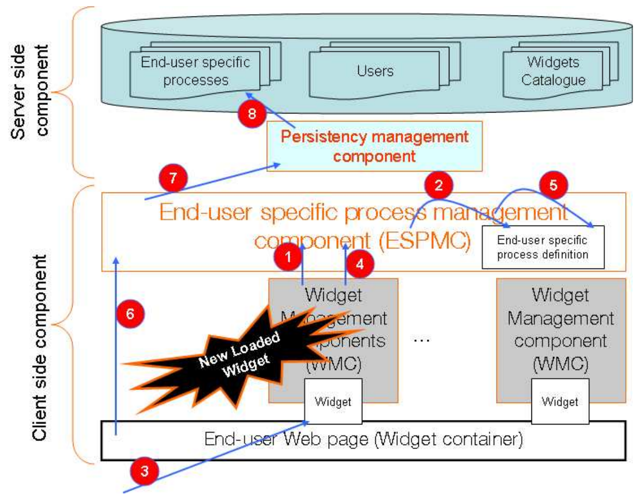

Server side component
End-user specific processes
Users
Widgets Catalogue
8
Persistency management component
7
2
5
End-user specific process management component (ESPMC)
End-user specific process definition
1
4
6
Widget Management components (WMC)
New Loaded Widget
…
Widget Management component (WMC)
Widget
Widget
Client side component
End-user Web page (Widget container)
3

Fig. 6. Process Creation.

D. *Execution of the End-user Specific Part of a Business Process*

In this section we detail how we enable the execution of an end-user specific business process within the Widget aggregator. As end-user specific processes are defined as a set of links, we detail more precisely the execution of a single link. Figures 7 and 8 illustrate respectively the execution of an automatic link, and a semi-automatic link.

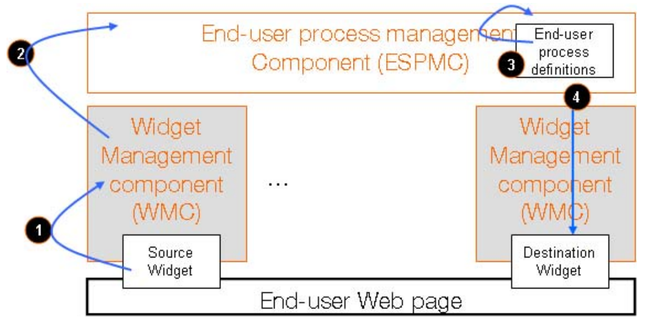
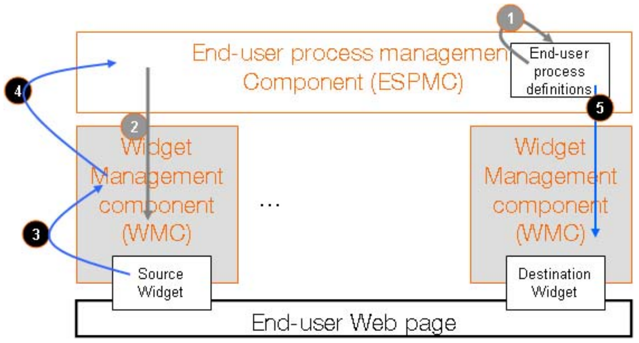

2
End-user process management Component (ESPMC)
End-user process definitions
3
4
Widget Management component (WMC)
…
Widget Management component (WMC)
1
Source Widget
Destination Widget
End-user Web page

Fig. 7. Automatic link execution.



End-user process management Component (ESPMC)
End-user process definitions
Widget Management component (WMC)
…
Widget Management component (WMC)
Source Widget
Destination Widget
End-user Web page
1
2
3
4
5

Fig. 8. Semi-automatic link execution.

An automatic link is executed when a Widget generates the data (e.g. *hCard*) and/or event (e.g. incoming call) that correspond to the link. Thus, for each data or event generated within the Widget, the ESPMC is notified (Step 1 and 2 in Figure 7). Then, the ESPMC checks (Step 3) if the data or the event corresponds to an automatic link within the end-user specific process definition. If that is the case, ESPMC retrieve from the source Widget the actual data (Step 4) and invokes the destination Widget (Step 5).

Semi-automatic links execution is performed during two phases: during the loading phase of the Widgets, and during the execution phase. During the loading phase, the ESPMC gets from the process definitions all semi-automatic links (step 1 in Figure 8). Then, for each link, it informs the corresponding source WMC to insert a UI element (e.g. an icon, or an HTML link) to the Widget. This UI element enables the end-user to launch the execution of the corresponding link (e.g. by clicking on the icon) (step 2). During this step, the WMC is also informed about the data that are required by each link.

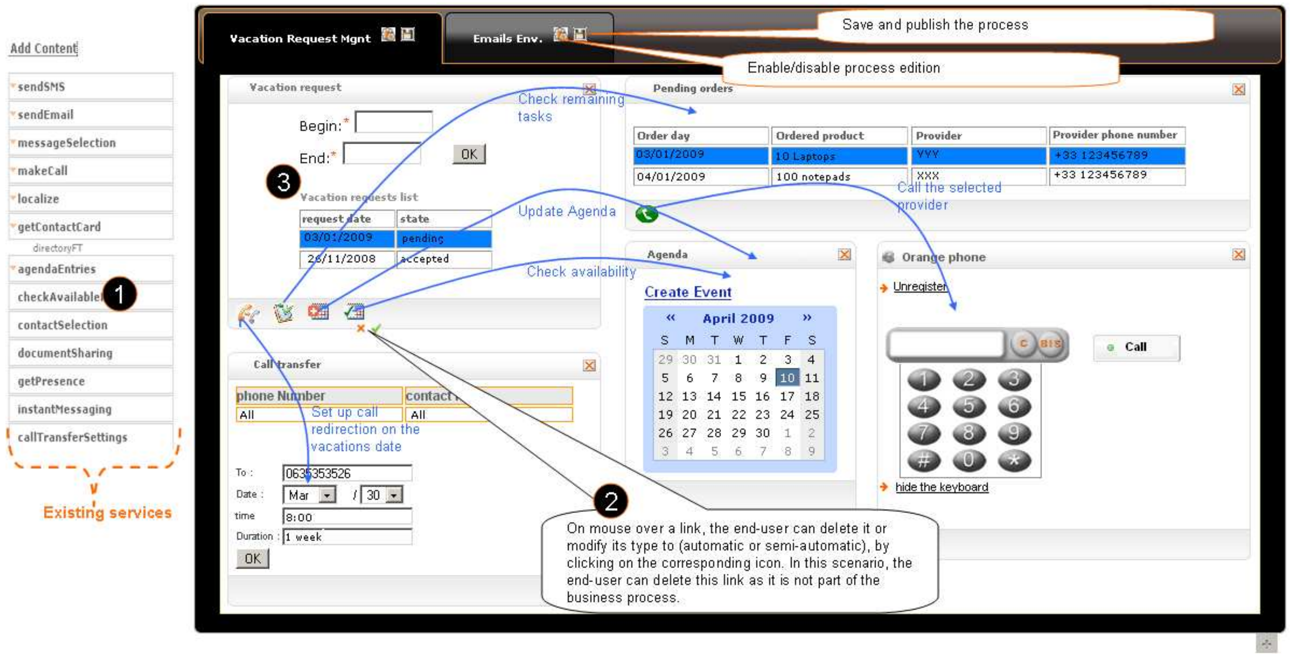
At the execution phase, when an end-user launches the execution of the link (clicks on the icon), the WMC retrieves from the Widget the values of the data that are required by this link, and transmits them to the ESPMC, with the corresponding link identifier (step 3 and 4). Thereafter, the ESPMC gets the destination Widget and invokes it (Step 5).

V. Framework Validation

The prototype we have detailed in the previous section has been experimented within Orange Labs among marketing and IT team. The feedback is unanimous: the functionality of running business processes within the Widget aggregator was appreciated, and the new approach we introduce for specifying business processes has been successfully tested by users without computing skills.

In this section, we illustrate the prototype through the scenario we provided in Section 2 (vacation request business process). Figure 9 shows the end-user environment that enables him/her to create and execute a personalized vacation request business process. To do that, three actions are required: (1) loading Widgets, (2) specifying an end-user process, and (3) executing the end-user process.

In this Figure, the end-user has already loaded the "Call Transfer" Widget, the "Pending Orders" Widget, the "Agenda" Widget, the "Telephony" Widget, and the "Vacation Request" Widget. The "Vacation Request" Widget implements the common part of the business process that performs a vacation request. In this illustration (Figure 9) all Widgets are visible to the end-user, but actually the end-user can deactivate them; the deactivated Widgets are executed only when the corresponding step in the process is reached.



Vacation requests list

request date	state
03/01/2009	pending
26/11/2008	accepted

Pending orders

Order day	Ordered product	Provider	Provider phone number
03/01/2009	10 Laptops	YYY	+33 123456789
04/01/2009	100 notepads	XXX	+33 123456789

S	M	T	W	T	F	S
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Call transfer

phone Number	contact
All	All

To : 0635353526
Date : Mar / 30
time 8:00
Duration : 1 week

On mouse over a link, the end-user can delete it or modify its type to (automatic or semi-automatic), by clicking on the corresponding icon. In this scenario, the end-user can delete this link as it is not part of the business process.

Fig. 9. Framework illustration.

As the end-user loads the Widgets into his/her environment, a "mesh" process has been automatically created. This process creates links between all Widgets that are loaded, according to semantic matching between their inputs and outputs. If the user wants to create a link to a Widget which is not loaded yet, he/she must at first load it.

The links between Widgets are represented by an HTML element that enables the end-user to modify the definition of the "mesh" process. He/she can easily delete an undesired link, or modify its type. For instance, as part of the "mesh" process, a link has been created between the "Vacation Request" Widget and the "Agenda" Widget, which enables the end-user to check his/her availability on a date generated by the "Vacation Request" Widget. However, in the end-user business process defined in Section 2, this link is not included as part of the end-user business process. Consequently, the end-user may want to delete it. This is performed through an intuitive interface element illustrated in Figure 9 (zone 2).

Once the end-user has finished defining his/her business process, he/she can use it by executing Widgets within his/her environment. He/she can also publish it in order to enable other end-users to load and use it.

When executing the business process, the end-user will retrieve exactly the same interface illustrated in Figure 9; except that the process edition is disabled. In other words, the end-user can not delete, or modify the type of, a link. In the example illustrated in Figure 9, the end-user will have in his/her Widget aggregator the Widgets needed to efficiently perform the vacation request process. This includes firstly the common part of the business process, which is implemented within the "Vacation Request" Widget, and secondly the end-user specific part which is specified by the end-user himself/herself and executed as linked Widgets. As a consequence, when a response to a vacation request is positive, the Widget generates the corresponding event, the framework invokes automatically: the "Call Transfer" Widget (to set up a call redirection during the leave period), the "Agenda" Widget (to set up the end-user unavailability during the leave period), and finally, the framework displays the pending orders to the end-user (to enable him/her to accelerate the process and finish it before leaving). The "Pending Orders" Widget is also linked to the "Telephony" Widget by inserting an icon that enables the end-user to call a provider of a selected item.

VI. Positioning the Proposed Framework

As we illustrate in Figure 10, the framework we propose in this paper does not intend to replace current service orchestration tools, which facilitate significantly business process implementation. It enables the end-users to personalize a business process according to their needs. We have shown in this paper that business processes comprise two parts: a common part and an end-user specific part. The former is usually long-lived and responds to a popular need of end-users, whereas the latter is heterogeneous, dynamic, and end-user dependent. Therefore, service orchestration tools (such as BPEL4WS [6] and SPATEL [11]) and our proposal might co-exist in the same environment. The former automates the part of the process which is common to all end-users, and the latter enables the automation of the part of the process which is dynamic and specific to a limited number of end-users. Thus, in Figure 10, the end-user accesses to Business processes that are designed and implemented by developers (Widget A and B). He/she accesses other Widgets created from scratch (send email, phone, and Agenda). Finally, he/she can combine these Widgets according to processes which are specific to him/her.

Figure 10 also illustrates that our contribution is an enhancement made to current Widget aggregators in order to succeed within an enterprise context. Indeed, platforms like iGoogle [1] and Netvibes [2] do not support the end-user in achieving his/her business goals. In [13 and 14], we have proposed new mechanisms named Drag & Drop and Communication Manager that enable the services wrapped within Widgets to communicate with each other. Both mechanisms are characterized by automatically creating links between independent Widgets and thus enable ordinary end-users, without any development skills, at the run-time, to chain them. However, both mechanisms create links according only to inputs/outputs semantic matching, and unfortunately, they do not take into account the end-user business activities. The framework we propose in this paper however enables to personalize these links according to end-user business activities, modeled as processes.

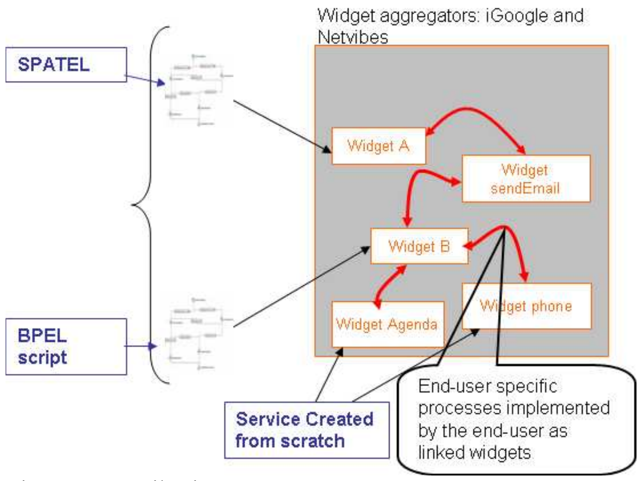

Widget aggregators: iGoogle and Netvibes
SPATEL
BPEL script
Widget A
Widget sendEmail
Widget B
Widget Agenda
Widget phone
Service Created from scratch
End-user specific processes implemented by the end-user as linked widgets

Fig. 10. Contribution statement.

VII. Conclusion

The paper presents a novel technique for implementing and easily integrating business processes using Widget paradigm. The driving idea is characterized firstly by wrapping each service within a Widget to promote human–to-machine interaction, and secondly, by providing end-users with a Widget aggregator framework that enables them not only to load the Widgets they need, but also to chain these Widgets with each other according to processes they have defined themselves. This work also proposes a novel approach that enables even ordinary end-users, without computing skills, to specify processes. This provides a solution for tackling the heterogeneity and the dynamicity of end-users processes and needs. Validated by the implementation and experimentation of a prototype within an enterprise context (Orange Labs), we hope by the present paper to foster further research and experimentations in other contexts (mass market or other organizations).

The present work makes significant advances in Widget aggregators as well as in the business process management. We enhance Widget aggregators by integrating natively a business process management tool, and we define and validate an intuitive approach for specifying the end-user part of business processes.

REFERENCES

[1] Google, http://www.google.com/ig

[2] Netvibes, http://www.netvibes.com

[3] N. Laga, E. Bertin, and N. Crespi, "A unique interface for web and telecom services: From feeds aggregator to services aggregator," *In ICIN 2008*, Bordeaux, France, 20-23 October 2008.

[4] R.S. Aguilar-Savén, "Business process modelling: Review and framework.," *International Journal of Production Economics*, 2004. Vol. 90 (2): p. 129--149.

[5] A. Stephen, "Introduction to BPMN", White, IBM Corporation, http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf

[6] T. Andrews, et al., "Business Process Execution Language for Web Services (BPEL4WS)", http://www.oasis-open.org/committees/download.php/2046/BPEL%20V1-1%20May%205%202003%20Final.pdf

[7] R. Khalaf, N. Mukhi, S. Weerawarana, "Service-Oriented Composition in BPEL4WS," *In Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungery, May 2003.

[8] G.Little, T. A. Lau, A. Cypher , J. Lin, E. M. Haber, and E. Kandogan, "Koala: capture, share, automate, personalize business processes on the web," *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA, April 28 - May 03, 2007). CHI '07. ACM, New York, NY, 943-946.

[9] R. S. Sadasivam, "An Architecture Framework for Composite Services with Process-Personalization," Doctoral Thesis. UMI Order Number: AAI3301404., University of Alabama at Birmingham, 2007.

[10] E. Newcomer, "Understanding Web Services: XML, Wsdl, Soap, and UDDI" *Addison, Wesley*, Boston, Mass., May 2002.

[11] M. Belaunde, and P. Falcarin, "Realizing an MDA and SOA Marriage for the Development of Mobile Services," *In Proceedings of the 4th European Conference on Model Driven Architecture: Foundations and Applications* (Berlin, Germany, June 09 - 13, 2008). I. Schieferdecker and A. Hartman, Eds. Lecture Notes In Computer Science, vol. 5095. Springer-Verlag, Berlin, Heidelberg, 393-405.

[12] J. Soriano, D. Lizcano, M. Cañas, M. Reyes, and J.J. Hierro, "Fostering innovation in a mashup-oriented enterprise 2.0 collaboration environment," *System and Information Science Notes, SIWN International Conference on Adaptive Business Systems*, Chengdu, China, July, Vol. 1, No. 1, pp.62-69.

[13] N. Laga, E. Bertin, N. Crespi, "A web based framework for rapid integration of Enterprise applications," *In the ACM International Conference on Pervasive Services,* Imperial College, London, UK, July 13-17, 2009.

[14] N. Laga, E. Bertin, N. Crespi, "Building a user friendly service dashboard: Automatic and non-intrusive chaining between widgets," *In the 2009 IEEE congress on Services,* Los Angeles, California, USA, July 6-10, 2009.3

[15] W3C, http://www.w3.org/TR/2007/WD-widgets-reqs-20070209/

[16] M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, and B. J. Krämer, "Service-oriented computing research roadmap," *in Dagstuhl Seminar Proceedings* 05462, April 2006.

[17] R. Khare, "Microformats: the next (small) thing on the semantic Web?," *Internet Computing, IEEE* , vol.10, no.1, pp. 68- 75, Jan.-Feb. 2006

[18] K. Stolley, "Using Microformats: Gateway to the Semantic Web," *IEEE Transactions on Professional Communication*, vol.52, no.3, pp.291-302, Sept. 2009.

[19] J. Gehtland, D. Almaer, and B. Galbraith, "Pragmatic Ajax: A Web 2.0 Primer," *Pragmatic Bookshelf*, 2006

[20] IETF, RFC 4627, http://www.ietf.org/rfc/rfc4627.