A Mobile End-User Tool for Service Compositions *

Ignacio Mansanet, Victoria Torres, Pedro Valderas, and Vicente Pelechano

Universitat Politècnica de València, Spain
Pros Research Center
{vtorres,imansanet,pvalderas,pele}@pros.upv.es

Abstract. With the advent of Web 2.0 and the massive adoption of smartphones, end-users are more keen to actively participate in the creation of content for the Internet. In addition, the big amount of data that the Internet of Things can bring to it, establishes an ideal framework to allow end-users not just consuming data but also creating new value-added services. To achieve this, in this work we propose the development of an end-user tool for smartphones capable of composing services that are available in the Internet.

1 Introduction

Since 2004 where the term Web 2.0 was popularized, the role played by end-users in the development of the web has dramatically changed. Since then, the availability of easy-to-use tools such as the ones developed within social networks, wikis, CMSs, etc. have promoted the interaction and collaboration of end-users to generate new content.

With the advent of the Internet of Things (IoT), a big amount of objects can be virtually represented in the Internet (by 2020 Gartner estimates nearly 26 billion of devices on the IoT[2]). These objects will offer data about their state; e.g., temperature, lightning, and sound pressure level will be provided by thermoters, lightning detectors, and sound meter levels respectively. These data are usually provided by web services which collect them by querying the corresponding source (i.e., the specific device or the database that stores them). Even though these services can be used individually, their composed usage has the potential to create new value-added services. However, the creation of such compositions requires users with programming background since existing composition environments (such as Intalio, Activiti, Signavio or Bonita BPM) and service composition languages or notations (such as Petri nets, EPC, YAWL, BPMN or UML Activity Diagrams) are not yet targeted to ordinary users without such skills.

* This work has been developed with the support of the Spanish Ministry of Economy and Competitivity IPT-2012-0839-430000 and the Valencian Government ACOMP/2014/186 and financed jointly with ERDF

[2] http://www.gartner.com/newsroom/id/2636073

Moreover, existing environments to create such compositions have been traditionally designed to be used from desktops or laptops. However, the type of devices that end-users may use are not longer of this type but mobile devices such as tablets or smartphones. According to Gartner[3], in the second quarter of 2013 worldwide smartphone sales to end-users reached a total of 225 million units. Compared to the second quarter of 2012, this figure represents an increase of 46.5 percent. Furthermore, worldwide smartphones sales have overtaken for the first time feature phone sales, constituting the 51.8 percent of mobile phone sales. All these figures confirm the smartphone mass adoption by end users. According to this evidence, make sense to think that end-users will use such devices not just for consuming data and services but also to create new ones either from scratch or from content that is already made available in the Internet.

Within this context the current work presents an end-user tool for the construction of service compositions by using mobile devices. To this end, the development of this tool has been inspired by change patterns and techniques for end-user development (EUD). This work is being developed within the context of SITAC (Social Internet of Things, Apps by and for the Crowd), an ITEA 2 project aimed at the convergence of the IoT and Social Networks to enable a better interaction among humans and devices.

The remainder of the paper is structured as follows. Sect. 2 presents a scenario that illustrates the practical application of the presented end-user tool. Then, Sect. 3 introduces the foundations of the tool, which are change patterns and end-user development. Sect. 4 explains in detail the design and implementation of the end-user tool. Sect. 5 presents some related work. Finally, Sect. 6 concludes the paper and provides insights of directions for future work.

2 Running Example

In order to exemplify the proposal, we present an example that will be used through the paper. The scenario describes a day in John's life. John is a 22 years old university student who is actually enrolled in his 4th year of studies. The university where John is studying offers some IoT services which provide the university community with information about university facilities (i.e., library availability, parking places availability, events celebrated in a specific date, etc.). During the exams period, John likes to go to the university library to study and meet his classmates. To do so, John is using the university facilities as follows. Before leaving home, John searches for libraries where there are still available seats and for bike parks closeby with free space for his bike. Based on the results of these searches John selects a library and a bike park and book places for him and his bike. In addition, while performing the booking, John notifies his classmates about the library where his is going to stay publishing this information via his Facebook account. Fig. 1 shows the BPMN process that describes all the tasks performed by John. Since John does these tasks every

[3] http://www.gartner.com/document/2573119

morning during this period, he wants to define a composition service linking all these services properly.

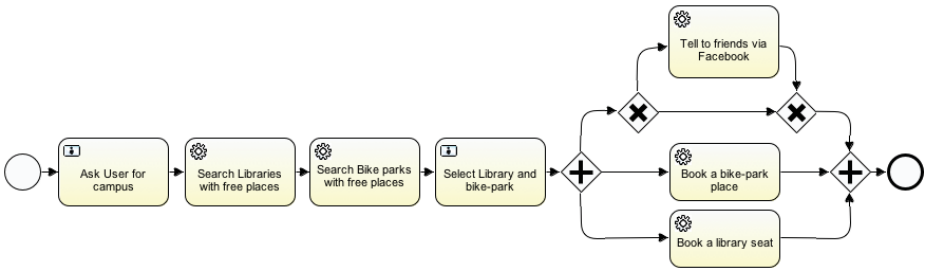

Fig. 1. Service composition for the running example specified in BPMN

3 Foundations

This section introduces the two research fields in which this work relies on. On the one hand, it presents the different techniques that have been developed within the EUD field. On the other hand, it presents change patterns and the advantages that these introduce when building process models, which are seen as descriptions of service composition at a higher level of abstraction.

3.1 End-user Development

The objective of EUD is to develop tools that can be successfully used for different types of users, i.e., novices and experts. While novices should find these tools easy to use, not be intimidating and giving them the enough confidence to succeed, experts should be able to work on sophisticated and completed solutions.

Many techniques have been developed in EUD for the development of applications. These techniques include programming-by-example, visual or textual languages. Programming-by-example is a programming technique where the user creates an application by performing the steps as an example. However, it is not easy to create the final application using this technique in isolation, since this do not allow end-users reviewing, testing, and debugging applications. To alleviate this problem, the programming-by-example technique is used in combination with visual or textual languages. For example, visual programming allows simplifying the development of applications since it provides users with environments that provide constructs in a higher level of abstraction, hiding the most tedious technical part. In this line of research, dataflow visual language is one paradigm that has been often applied. An example of this technique is Yahoo! Pipes (http://pipes.yahoo.com/) that provides an environment capable of aggregating, manipulating and mashing up content from the Web.

On top of these techniques there are recommendations or design guidelines aimed at helping in the development of such tools [1]. These recommendations deal with syntactic, semantic, and pragmatic aspects that need to be taken into consideration to develop effective end-user tools. Well-known design suggestions and patterns [2, 3, 4, 5] in the area of EUD include (1) the usage of the catalogue pattern, which offer end-users a set of predefined elements to face the selection barriers [3], (2) tabs which improve the learning and memorization of the steps to be performed and keeps end-users from missing important things [4], (3) textual descriptions of the results in order to facilitate its understanding, or (4) designing all the interfaces following a similar structure to facilitate user interpretation of the information.

In the EUD community, it is accepted that the use of wizards can save end-users from a potentially frustrating experience [4]. For this reason, for the development of our end-user tool we rely on wizards following the design suggestions and patterns presented above. Wizards will guide users through a particular process with the objective of creating service compositions.

3.2 Change Patterns

Change patterns are high level abstractions aimed at achieving flexible and easy adaptations of business process [6]. These abstractions are defined in terms of high level change operations (e.g., the creation of a parallel branch) which are based on the execution of a set of change primitives (e.g., add/delete activity). As opposed to change primitives, change pattern implementations typically guarantee model correctness after each transformation [7] by associating pre-/post-conditions with high-level change operations. In process modeling environments supporting the correctness–by–construction principle (e.g., [8]), usually process modelers only have those change patterns available for use that allow transforming a sound process model into another sound one. For this purpose, structural restrictions on process models (e.g., block structuredness) are imposed. In addition, a correct usage of change patterns allows speeding up the creation of the composition (while the construction of Fig. 1 requires 31 steps with change primitives, just 12 steps are needed when using change patterns).

In this work we take as basis from [6] a small subset of change patterns to allows users to create all main control-flow constructs (i.e., sequences, parallel branches, conditional branches, and loops). This set includes: AP1 (Insert Process Fragment) with its two variants, i.e., Serial and Parallel Insert, AP2 (Delete Process Fragment), AP8 (Embed Fragment in Loop), and AP10 (Embed Process Fragment in Conditional Branch). For illustration purposes, Fig. 2 shows the changes introduced by the application of pattern AP8 into S to obtain S' where activity B is embedded in a loop.

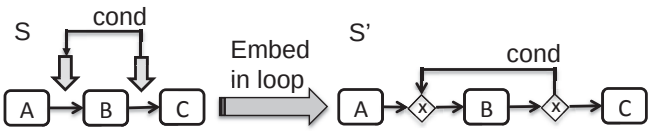

Fig. 2. Pattern AP8: Embed Process Fragment in Loop

4 End-user Tool for Service Composition

This section presents the Domain Specific Language (DSL) that has been designed to implement the end-user tool for service composition. While its abstract syntax has been defined based on the change patterns introduced in Sect. 3.2, its concrete syntax has been defined based on the wizard concept.

4.1 Domain Specific Language: Abstract Syntax

Fig. 3 shows the metamodel that defines the DSL that allow specifying service compositions in a high level of abstraction. Inspired by the set of patterns presented in Sect. 3.2, the metamodel is made up of two main concepts which are *activities* and *fragments*. This facilitates their definition by end-users since they only have to understand these two concepts. *ServiceElements* are defined through the composite pattern, which facilitate the creation of user interfaces that help end-users to focus on the definition of either the service main sequence or the content of a fragment, individually. The *previousElement* relationship between *ServiceElements* allows establishing the sequence order between *activities* and *fragments*. We thought that it would be easier for end-users without notions of process modelling to create nodes associated to a previous one than explicitly create flow elements. In addition, this sequence order implicitly defines the start and end of the composition, being the first and last *activity* or *fragment* the start and the end of the composition respectivelly. Finally, *services* and *parameters* are complemented with a description attribute, whose main goal is to provide end-users with a textual description that helps them to understand the semantics of a service or a parameter.

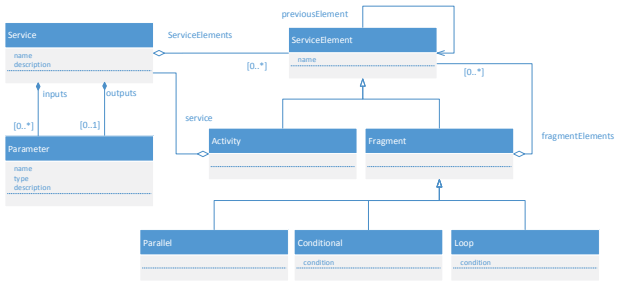

Fig. 3. Domain Specific Language

In order to better understand the concepts of this metamodel, Fig. 4 identifies them in the process used as a running example. This process is composed of a sequence of four activities followed by a parallel fragment. In turn this fragment is made up of a conditional fragment and two activities being executed in parallel to it.

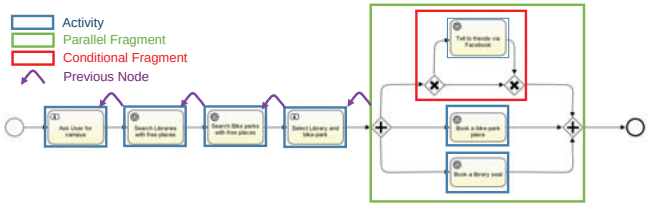

Fig. 4. DSL concepts applied to the running example

4.2 Domain Specific Language: Concrete Syntax

In this section, we introduce the concrete syntax of the end-user tool, which is based on the use of wizards. The primitives used in the wizards are in accordance with the abstract syntax presented above. The implementation of this wizard has been performed by applying styles and designs for mobile devices. To achieve this, we have used the jQuery Mobile framework[4]. In particular, wizards have been defined considering the following well-known design suggestions and patterns (see Fig.5).

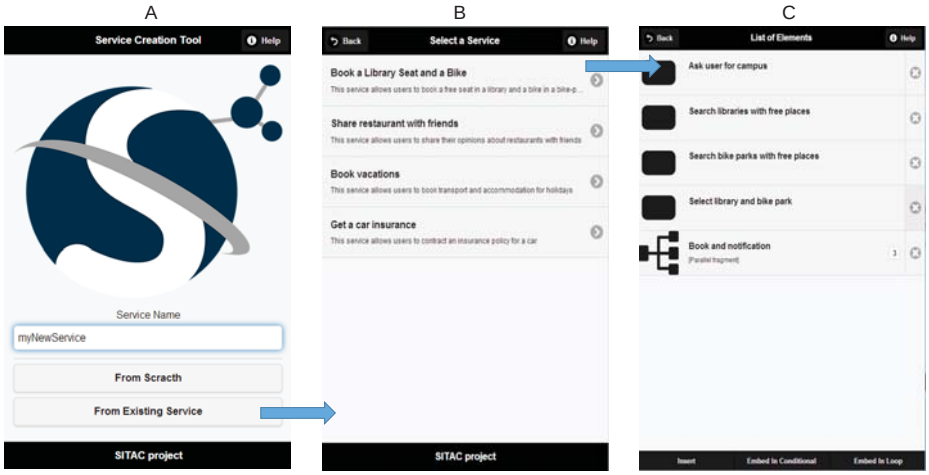

Fig. 5. Wizard to create a service composition from an existing service

[4] http://jquerymobile.com

Catalog pattern. We use the catalog pattern to offer a set of predefined elements, services in our case, to end-users. A detailed description of these elements is presented in Fig. 6A. From left to right, each service element (activity or fragment) is described by three graphical components: (1) an icon, which allows identifying the type of element being represented (activity or fragment, and in case of a fragment, whether this is parallel, conditional, or loop); (2) the name of the element; and (3) the number of elements included in a fragment (just for fragments). Note that the order in which tasks and fragments appear in this interface indicates the sequence order between them.

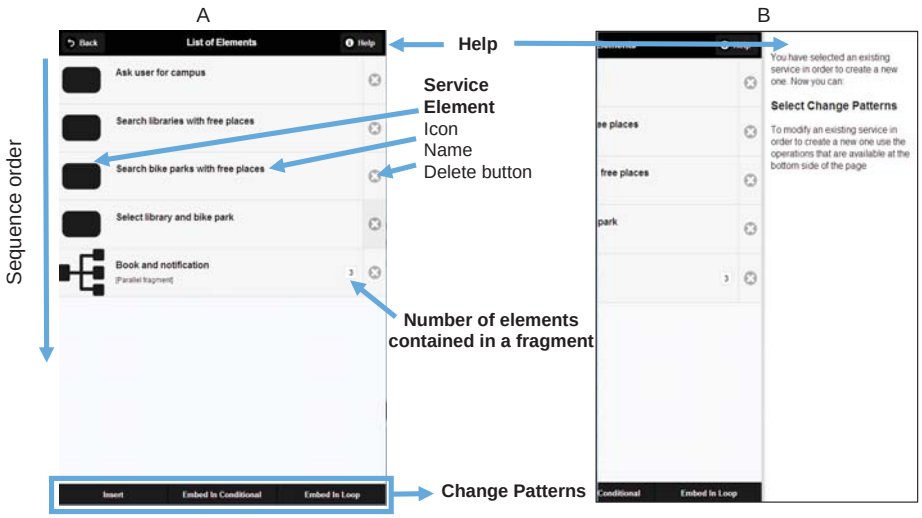

Fig. 6. Details of a service composition

Tabs. We guide users by using tabs that indicate what they have already done and what they can do along the wizard. As Fig. 6B shows, each user interface has a Help button that allow users to access a tab that explains the actions that end-users have already done and the actions that they can do from this point.

Change patterns. To create service compositions, end-users can make use of any of the available change patterns that are implemented in the tool. These patterns can be access from the buttons that are provided in the *list of Elements* user interface (cf. Fig. 6A) at the bottom side of the page, i.e., Insert (AP1–Insert Process Fragment), Embed in Conditional (AP10–Embed Process Fragment in Conditional Branch), and Embed in Loop (AP8–Embed Fragment in Loop)). Differently from these patterns, the Delete pattern (AP2–Delete Process Fragment) is directly used from the corresponding element by clicking on the x button placed at its right-hand side (cf. Fig. 6A). If end-users select to embed an element in a conditional branch or in a loop pages A and B in Fig.7 are accessed, respectively. In both cases, end-users must indicate a name and a condition. In

the case of a conditional branch, the elements that must be executed when the condition is satisfied must be selected. In the case of a loop fragment, end-users must select the first and last element in the loop in order to indicate that after the last element, if the condition is satisfied, the flow of the service must go to the first element. Note that both pages show a textual description of the result at the bottom side.

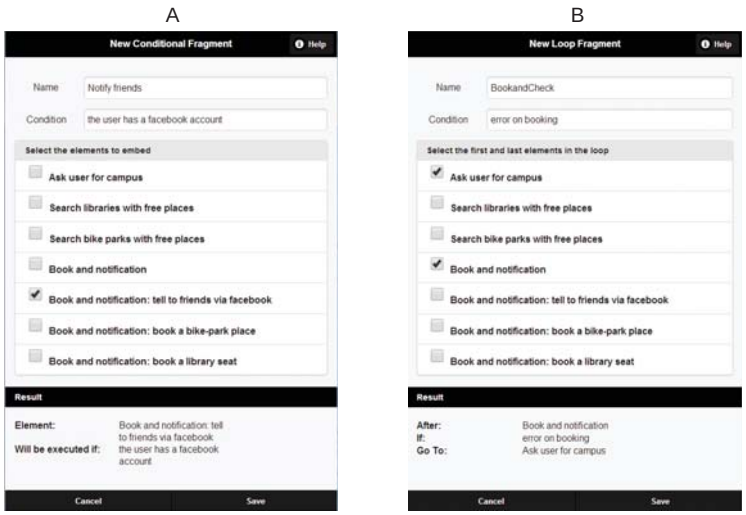

Fig. 7. Embeding elements in a conditional branch or loop

When end-users select to insert a service element the page in Fig. 8A is shown. From this page, end-users can select to insert an activity or a parallel fragment. If they select the first option, the page shown in Fig. 8B is accessed. Then, end-users can create and activity by indicating a name, the previous element, and a predefined service. The predefined service is selected from the catalogue of existing ones that is supported by the tool. Note that a textual description of the result of this insert action is shown at the bottom side of the page. To finish this step, the inputs of the predefined service must be configured. To do so, end-users can access the page in Fig. 8C, which allows them to indicate if inputs must be introduced by users or must be defined from the output of an already defined service element (such as in Fig. 8C). In this last case, end-users can select the outputs of a service element that will be used as inputs.

If end-users select to insert a parallel fragment page in Fig. 9A is shown. From this page, a parallel fragment can be created by indicating a name, a previous element, and the elements that must be executed in a parallel way. Note how a textual description of the results is shown at the bottom side of the page. When a parallel fragment is created it is inserted in a sequence of the service as it is shown in Fig. 9A. From this sequence, the fragment can be selected in order to access the page in Fig. 9C, which show the elements that it includes. From this

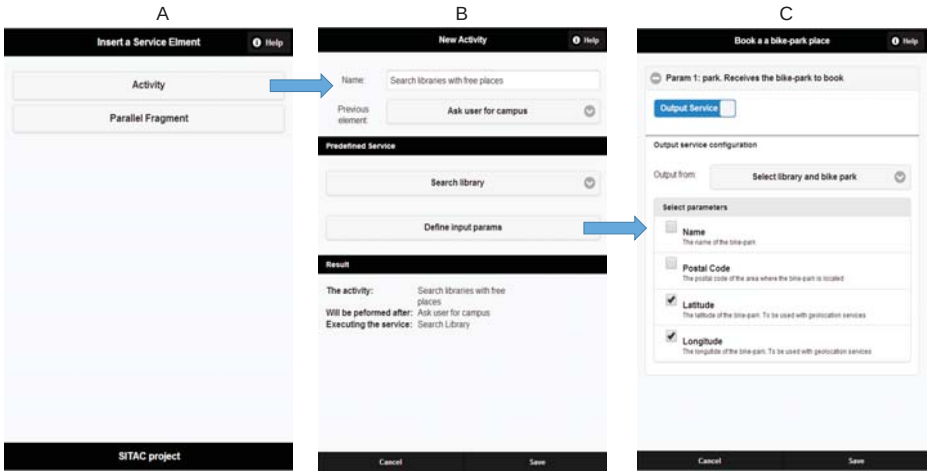

Fig. 8. Inserting an activity

page, the change patterns (Insert, Embed in Loop, Embed in Conditional) can be applied to the content of the fragment.

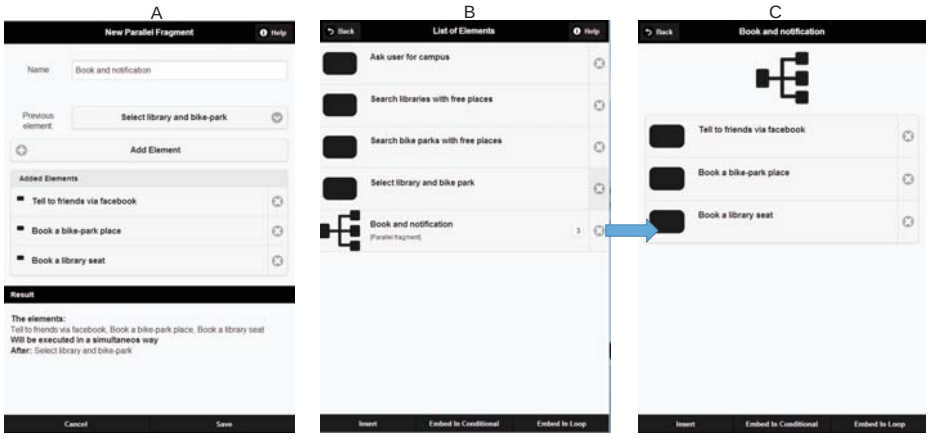

Fig. 9. Inserting a parallel fragment

Textual descriptions. Actions done by end-users are complemented with textual descriptions of the results in order to facilitate its understanding.

Similar structures. All the interfaces (as Fig. 5-9 show) share a similar structure to facilitate user interpretation of the information.

5 Related Work

The mass adoption of mobile devices by end-users together with the amount of services available in Internet has motivated the development of research of Mobile mashups, a field aimed at the development of tools that allow end-users the aggregation, manipulation, and mashuping content from around the web using such devices.

Similarly to our end-user tool, existing research in this field proposes the development of tools based on well-known EUD techniques. By using different Visual languages, these works propose authoring tools aimed at the construction of applications based on services offered by the device at hand or services coming from external services. For example, [9] proposes SCK (Service Creation Kit), an authoring prototype tool for the development, deployment and execution of microservices (small applications that allow users to obtain and provide information from fellow users). Antoher example aimed at the construction of mobile applications is [10]. Here authors propose the generation of mobile applications by specifying the sequence of actions needed to model the required application. Finally, in [11] authors propose an EUD framework based on model-driven development techniques. This framework is aimed at the design and automatic generation of mobile mashups.

We argue that the main difference between our work and all these proposals is that the design of our end-user tool relies on change patterns. As we explained in Sect 3.2, the two major advantages of using change patterns are that ensure correctness-by-construction and that they speed up the construction process. In addition, these patterns allow us to provide end-users a more powerful tool that allows creating service compositions using not just sequences or paralallel branches but also embedding fragments into conditional and loop structures.

6 Summary and Future Work

In this work we have presented an EUD tool aimed at the composition of services by means of a smpartphone. The main objective of this tool is to provide end-users with all the necessary functionality to create service compositions based on the services provided by the SITAC platform. This functionality has been designed based on a specific set of change patterns and implemented using recommendations and guidelines from the EUD field. However, such service compositions are not yet executable compositions. In fact, end-users did not have to deal with technological details (e.g. BPMN, REST, XML) that are needed to execute such compositions. To achieve this, the end-user tool will be complemented with a set of components (e.g., data mappings and transformations, invoking REST services, etc.) that are needed to execute them in a process engine. Moreover, the idea is to automate as much as possible the tasks performed by these complementary components so they remain transparent to the end-user.

As future work we plan to assess, by means of the Technology Acceptance Model (TAM), end-users' perceived usefulness and perceived ease of use. In addi-

tion, these subjective measures will be contrasted with users' actual performance with think-aloud sessions which will allow us to analyze the problem-solving processes followed by end-users during the usage of the tool.

References

1. Repenning, A., Ioannidou, A.: What makes end-user development tick? 13 design guidelines. End User Development **9** (2006) 1–41
2. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. **37** (2005) 316–344
3. Ko, A.J., Myers, B.A., Aung, H.H.: Six learning barriers in end-user programming systems. In: Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing. VLHCC '04, Washington, DC, USA, IEEE Computer Society (2004) 199–206
4. van Welie, M., Traelig;tteberg, H., Trtteberg, H.: Interaction patterns in user interfaces. In: Proc. Seventh Pattern Languages of Programs Conference: PLoP 2000. (2000) 13–16
5. Mick, C.P., Roger, T., Frederick, C.G., Scott: What They See Is What We Get: Response Options for Web Surveys. Social Science Computer Review **22** (2004) 111–127
6. Weber, B., Reichert, M., Rinderle, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. Data and Knowledge Engineering **66** (2008) 438–466
7. Casati, F.: Models, Semantics, and Formal Methods for the design of Workflows and their Exceptions. PhD thesis, Milano (1998)
8. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Comp Scie - R&D **23** (2009) 81–97
9. Danado, J., Davies, M., Ricca, P., Fensel, A.: An authoring tool for user generated mobile services. In Berre, A., Gmez-Prez, A., Tutschku, K., Fensel, D., eds.: Future Internet - FIS 2010. Volume 6369 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2010) 118–127
10. Cuccurullo, S., Francese, R., Risi, M., Tortora, G.: Microapps development on mobile phones. In Costabile, M., Dittrich, Y., Fischer, G., Piccinno, A., eds.: End-User Development. Volume 6654 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 289–294
11. Cappiello, C., Matera, M., Picozzi, M.: End-user development of mobile mashups. In Marcus, A., ed.: Design, User Experience, and Usability. Web, Mobile, and Product Design. Volume 8015 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 641–650