Department of
Information Engineering
and Computer Science
DISI

UNIVERSITY
OF TRENTO - Italy

DISI - Via Sommarive 14 - 38123 Povo - Trento (Italy)
http://www.disi.unitn.it

Conceptual Design and Evaluation of WIRE: A Wisdom-Aware EUD Tool

Antonella De Angeli, Alberto Battocchi, Soudip Roy Chowdhury, Carlos Rodriguez, Florian Daniel, and Fabio Casati

Department of Information Engineering and Computer Science
University of Trento
Via Sommarive, 14 – 38123 Povo, Trento (Italy)
{antonella.deangeli, alberto.battocchi, soudip.roychowdhury, carlos.rodriguez, florian.daniel, fabio.casati}@unitn.it

March 2011

Technical Report # DISI-11-353
Version 1.0

Accepted for publication at IS-EUD 2011
Third International Symposium on End-User Development
June 7-10, 2011

Conceptual Design and Evaluation of WIRE: A Wisdom-Aware EUD Tool

Antonella De Angeli, Alberto Battocchi, Soudip Roy Chowdhury, Carlos Rodriguez, Florian Daniel, Fabio Casati

Department of Information Engineering and Computer Science
University of Trento
Via Sommarive, 14 – 38123 Povo, Trento (Italy)
{antonella.deangeli, alberto.battocchi, soudip.roychowdhury, carlos.rodriguez, florian.daniel, fabio.casati}@unitn.it

Abstract. This paper presents the evaluation of the conceptual design of WIRE, a EUD tool for service-based applications. WIRE exploits community composition knowledge harvested from existing programs defined by other developers in the same domain. Such knowledge can assist less skilled developers in defining the composition they need, allowing them to go beyond their individual capabilities. The assistance comes in the form of interactive contextual advices proposed during the definition of composition logic. This idea was evaluated with 10 semi-structured interviews with University accountants. A rich set of information was elicited by means of several probes, including examples of contextual helps, commercial EUD tools, and scenarios in the form of positive and negative user stories. Results informed the definition of a set of requirements for WIRE, and fostered a critical reflection on possibilities and limitations of the general framework of EUD.

Keywords: Contextual help; EUD requirements; Wisdom-aware development; Interactive Advice; Mashups.

1 Introduction

Although the requirement for 'support and help' clearly emerges from user research on EUD for web services [1, 2], little is available to satisfy this need. There are currently two main approaches to enable less skilled users to develop programs: development can be eased by *simplifying* it (e.g., limiting the expressive power of a programming language) or *reusing knowledge* (e.g., copying and pasting from existing algorithms). Among the simplification approaches, the workflow and Business Process Management (BPM) community was one of the first to propose that the abstraction of business processes into tasks and control flows would allow also less skilled users to define their own processes. Yet, according to our opinion, this approach achieved little success and modeling still requires training and knowledge. The advent of the service-oriented architecture (SOA) substituted tasks with services, yet composition is still a challenging task even for expert developers [1, 2]. The reuse approach

is implemented by program libraries, services, or templates (such as generics in Java or process templates in workflows). It provides building blocks that can be composed to achieve a goal, or the entire composition (the algorithm – possibly made generic if templates are used), which may or may not suit a developer's needs.

This paper presents the conceptual evaluation of WIRE, a *WIsdom-awaRE development environment* for exploiting the benefits of simplification *and* reuse. WIRE targets process-oriented, mashup-like applications, whose development and execution can be provided as a service via the Web and that are characterized by relatively simple composition logic and complex tasks or components. This class of programs provides the benefit of (relative) simplicity and a sufficient information base (the components) to learn and reuse composition knowledge. The idea is to *learn from existing compositions* (or, in general, *computations*) and provide this knowledge in the form of interactive advices to developers while they compose applications. The paper is organized as follows: Section 2 summarizes current approaches to assist users in development tasks; Section 3 introduces WIRE; Section 4 describes the evaluation and Section 5 concludes presenting implications for contextual help for EUD.

2 State of the Art

Several approaches are available to assist users in development tasks. In Programming by Demonstration [3], users are given examples of the process to be automated and based upon these examples the system auto-completes the remaining part of the process. Pattern-based Development [4] relies on a library of predefined patterns that represents good development practices. Goal-oriented approaches [5] assist developers by recommending plans that stem from user-specified goals. In semantic-based approaches [6], development ingredients are annotated to generate recommendations based on semantic matching and similarity techniques. Knowledge-discovery approaches [7] exploit a repository of previously developed applications in order to derive patterns that capture the steps performed solving similar problems in the past. All these approaches provide recommendations based upon explicit user inputs in the form of text queries, goals or partial application specification that convey the user intention. However, interactive recommendation systems can be improved by taking into consideration the composition status and the actions performed by users during development, as well as their preferences and skills, which is something that is not fully addressed by current approaches. Moreover, most of these approaches provide recommendations in the form of auto-completion, which has the limitation of masking the intermediate steps from the user, which in turn, results in loosing the control over the development.

In WIRE, we aim at discovering compositional knowledge by observing what other users have done in previous composite applications (i.e., applications that combine data, services and user-interfaces from multiple and possibly heterogeneous sources). We aim at delivering this knowledge in the form of interactive assistance, retrieved, filtered and ranked based upon context, user preferences and skills.

3 WIRE

To illustrate the challenges users face when using mashup environments, let us consider a typical scenario with Yahoo! Pipes[1], a composition environment that is currently presented on-line with words like 'intuitive' and statements like 'learn to build your own pipe in a few minutes'.

John is a soccer fan and an active blogger. He uses his blog to post latest news, articles, videos and updates from media sources and to discuss them with his friends. Keeping his blog updated requires a lot of manual work, such as content aggregation, filtering, and publishing. To automate it, John decides to use Yahoo! Pipes to build a simple pipe that: (1) sources a set of news feeds; (2) includes only the content related to soccer; (3) lists the news with their titles, and; (4) aggregates similar news from different sources under the same title. The pipe that implements the required feature is illustrated in Fig. 1. It is composed of five components. *Fetch Feed* gets the news from the publishing website (e.g., *feed://rss.soccernet.com/c/668/f/8493/index. rss*). *Fetch Page*, embedded inside the *Loop* component, fetches the page content from the feed output and extracts the content specified by 'Cut Content From and to' field. The *Unique* component merges the content of similar news, based upon their title (*item.title*) and groups them. Finally, *Pipe Output* represents the end of the pipe.

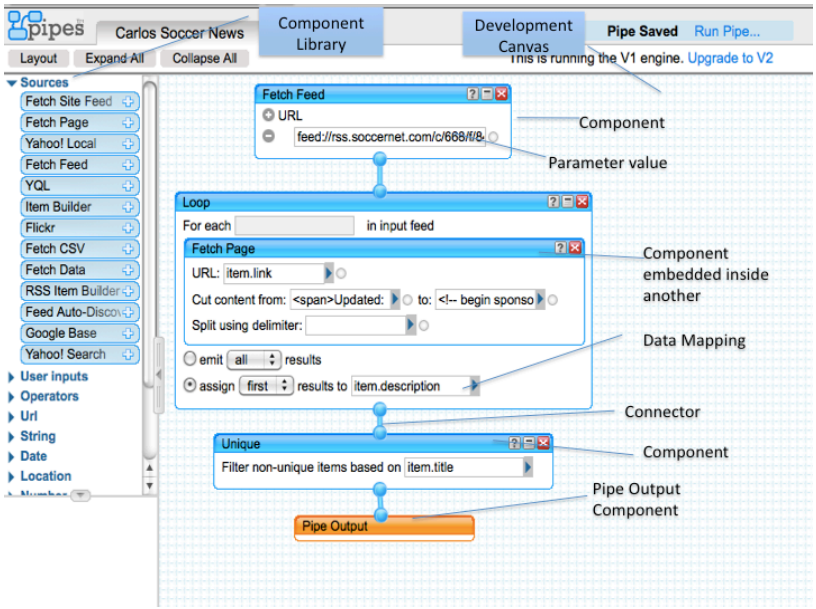

Fig. 1. Implementation of the example scenario in Yahoo! Pipes.

[1] http://pipes.yahoo.com/pipes/

The example shows that this apparently simple application involves development knowledge that goes beyond what typical end users have [8]. It requires knowledge about *programming models* (e.g., how to use components, connectors and data flows), *conventions or standards* (e.g., the structure of URLs), or simply about the right *terminology*. The intuition in WIRE is that this knowledge can be captured by *patterns* and reused as advices, to assist end users in their task. WIRE aims to leverage on the wisdom of the crowd for discovering and reusing *computational knowledge*. In [9] we introduced a detailed conceptual model of WIRE, which we summarize here for the better understanding of the ideas it encompasses. The conceptual model of WIRE can be divided into two parts: (1) the composition advices (the types of composition knowledge patterns) and (2) the triggering logic (when to deliver a given advice). *Advices* represent development recommendations, they can be of four types: *Complete* (e.g., a pipe with a missing connector), *Substitute* (e.g., a component in a pipe), *Highlight* (e.g., a connector between two components) or *Filter* out (e.g., a component from a pipe). Advices provide recommendations in the form of patterns, such as Parameter Value patterns (i.e., possible values for the parameter of a component), Component Association patterns (i.e., which components frequently appear together in a composition), Connector patterns (i.e., possible connections between components), Data Mapping patterns (i.e., mapping between input and output), or Complex patterns (i.e., a combination of the previous patterns). As for the triggering logic, a *Trigger* can be regarded as a triplet that consists of an Action, an Object and a Partial Composition. An *Action* represents what the user is doing, an *Object* represents the element on which the action is performed, and the *Partial Composition* represents the context for giving the advice. An Action can be further specialized as *Select* (a connector), *Drag* and *Drop* (a component from a component library), *Connect* (two components), *Fill* in (a parameter), *Delete* (a component), and *Embed* (a component into another). Finally, an Object can be a *Connector*, *Component*, *Parameter*, or *Composition Fragment*.

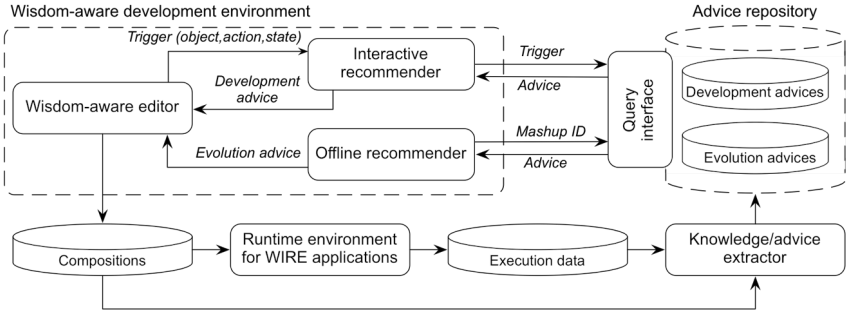

Fig. 2. High-level architecture of the wisdom-aware development (reported in [9]).

Fig. 2 presents the high level architecture that we envision for WIRE [9]. Development takes place in the *wisdom-aware environment*, where users can interact with a *wisdom-aware editor* for developing their applications. While a user builds his composition, a trigger may be fired based upon the actions that the user is performing in the editor. The triggering condition (object, action, partial composition) is sent to the *Interactive recommender*, which queries the *Advice repository* for a development

advice that matches the triggering condition. The advice is then sent back to the Wisdom-aware editor. The Advice repository is filled with composition patterns by the *Knowledge/Advice extractor*. It takes as input a repository of existing *Compositions* and leverages on data mining and statistical data analysis techniques to discover and harvest the patterns discussed previously. The retrieved advices are filtered, ranked, and delivered based on user profile data (e.g., the programming expertise of the user or his/her preferences over advice types). We omit the explanation of the *Offline recommender*, as such is less related to the object of this paper.

In summary, the aim of WIRE is to develop an environment that allows end users to build their own composite applications. While we have used Yahoo! Pipes to draw our examples, our goal is to apply the WIRE approach to our own mashup platform [10]. In what regards the reuse of knowledge, we do not base our approach on semantic content and technologies, artificial intelligence planning, or automated goal-driven composition. Instead, we aim at leveraging on the expertise of users expressed through what they do in practice (i.e., composing applications) and that is expressed in their compositions or mashups.

4 Evaluation

An evaluation of the conceptual design of WIRE was run in order to address benefits and limitations of the proposal at a very early phase of the design. The evaluation was conducted by means of semi-structured interviews.

4.1 Method

Participants. The interview was administered to 10 accountants (7 F, 3 M; mean age = 37,2 years of age), all of which were employees of the administration of a local university. No specific inclusion criteria were applied to the sample. None of the participants had a background in computer science. The choice of involving administrative accountants in the evaluation was motivated by the fact that they are non-technical users and that they represent the potential *real targets* of the system.

Procedure. Interviews were administered in a quiet room, in which only the participant and the experimenter were present, and were audio recorded for successive analysis. No time restrictions were applied and participants were informed that they had the right to withdraw at any time or avoid answering any of the questions. Participation to the interview, which lasted approximately one hour, was rewarded with a voucher of 15 Euro that could be spent on an online bookstore. The interview was divided in four sections.

Section A: IT knowledge with specific reference to SOA. This section aimed to understand the level of expertise that our sample had with computer systems, their use of computers during day-to-day work activities, the kind of software they mainly work with, and the level of acquaintance with web-services and mash-up application.

Moreover, we wanted to understand if and how participants were used to describe their work processes through graphical representations.

Section B: Automatization of repetitive and complex tasks. This section aimed at understanding which daily activities participants considered as repetitive or complex and which were the strategies or instruments, if any, that they used to reduce the workload deriving from repetitiveness or complexity. Then, participants were shown an example of automation of workflow created using Automator, an application developed by Apple for Mac OSX that allows implementation of workflows through the creation of batches of tasks. Tasks can be grouped into batches through drag-and-drop of action elements to a workflow space. After seeing the example of automation of the workflow, participants were asked two more questions, which are reported in Table 1.

Table 1. Questions included in Section B/2 of the Interview.

B/2	General questions regarding automatization of repetitive and complex tasks.
B.7	Can you please provide your comments about the application and the procedure of automatization you have just seen?
B.8	Imagine you have a strict deadline. In this case would you prefer to invest some of your time in the process of automatization of the tasks you are dealing with or would you follow a manual procedure, even if facing the risk of missing the deadline and making errors?

Section C: Computer-provided Help and advice. Section C targeted the difficulties that may emerge while using computers during day-to-day work activities and the strategies that people use for overcoming them. This section also aimed at understanding the attitude of participant towards computer-provided help and advice, with particular focus on automatic/contextual modalities that are commonly used to provide them. Section C was divided into two parts: part one (Tab. 2) contains three questions about difficulties and strategies to overcome them, and preference towards either automatic/contextual or on demand modalities for providing help and advice.

Table 2. Questions included in Section C/1 of the Interview.

C/1	Questions about computer-provided help and advice
C.1	(a) What kind of difficulties do you encounter while using computers during your habitual work activity? (b) How did you overcome these difficulties? (c) Where/how did you find help?
C.2	Which is the most effective help strategy among the one you reported?
C.3	Currently there are two main ways through which software provide help or advice to users: 1) following a request by the user (e.g. the user opens the "Help" menu); 2) Automatically (by interpreting the actions performed by users and guessing their objectives). Which of the two ways do you prefer? Why?

After answering the first set of questions, participants were shown a slideshow containing examples of automatic computer-provided help or advice. The presentation included: 1) *Automatic word completion* in the Google search box; 2) *Automatic friend suggestion* in Facebook; 3) *Automatic book suggestions* in Amazon (i.e. "fre-

quently bought together", or "customers who bought this item also bought"); 4) The function offered by web browsers that allows to *automatically save passwords; 5) Pop-up reminder windows* on calendars; 6) The *related videos* sidebar on YouTube. Participants were invited to comment each examples and were probed by three questions that related to advantages and drawbacks of each example, technical understanding on how the advice was created and perceived utility.

Section D: WIRE. Section D aimed at collecting opinions and suggestions about WIRE. We provided interviewees with a plus and a minus scenario reporting of an accountant who is using WIRE for automatizing the process of management of travel reimbursement. Following the methodology described in [11], both scenarios described the effects on work practices, which would be brought forward by WIRE on a new user. These effects were taken to the positive or negative extreme, to help users to think by themselves what consequences the new approach could have in their work practice. In the *Positive Scenario*, the accountant had a successful experience using WIRE, which helped him to save time and speed up repetitive work. So he decided to use it in the future and to share his work with other colleagues. In the *Negative Scenario*, the accountant encountered serious difficulties and eventually decided to go back to his traditional procedures for carrying out his work. Scenarios were presented with a counterbalanced order, meaning that five participants read the negative scenario first and the other five participants read the positive one first. The two scenarios are reported in the Appendix. After reading the two scenarios, interviewees were asked some specific questions about WIRE (Table 3).

Table 3. Questions included in Section D of the Interview.

D	Questions about the WIRE system
D.1	Do you think you would be interested in using WIRE?
D.2	Which advantages do you see in using this approach?
D.3	And what disadvantage do you see?
D.4	Within WIRE, advice and helps are created on the basis of behaviour and hints provided by people who previously used the system. Do you think that this kind of help is efficient in meeting your needs?
D.5	Would you prefer to have the advice automatically displayed or to receive them on request?
D.6	Would you like the system to inform you before doing tasks or during the task?

4.2 Results

Results are reported in different sections according to the thematic area addressed in the interview.

Sample Description. Almost all the participants learned to use a computer by themselves, through use at home and on the work place. Only 2 out of the 10 participants

attended specific courses that were aimed at providing the required knowledge on computer use for obtaining the European Computer Driving License (ECDL) certificate. Yet, participants in general reported considering their computer skills as good enough for dealing with the activities required by their job responsibilities. They mainly used programs for the productivity, in particular Microsoft® Office™, platforms for accountancy, e-mail clients and various web browsers. One of the participants had an education in electrical engineering.

IT knowledge with specific reference to SOA. The most part of the interviewees did not know what web services are, but all of them tried to provide an explanation about what they could be. Their definitions tended to encompass practical functionalities related to the meaning of the word 'service' in real life showing almost no understanding of the software engineering definition of the concept. Definitions provided by users varied from *websites for online banking, shopping, or music download*, to *web-based shared documents* (e.g. Google Docs), to *instruments for supporting collaboration* (e.g. Doodle), or software that help people to *improve their navigation skills*. None of the participant could provide a definition of mashup applications.

Only three participants reported to draw flowcharts to represent their work processes, while other two usually draw Gantt diagrams. These diagrams were usually initially sketched using pen and paper and then converted into electronic files using specific software or web-based applications for project management, or Microsoft® Excel™. One participant reported that he usually draws diagrams for describing the results of a process, rather than for planning it. Two participants declared that they have never used diagrams or any type of cognitive artifacts to describe their work.

Automatization of repetitive and complex tasks. All but one of the interviewees could readily report few examples of activities they deal with on a daily basis that they defined as repetitive. Three of the participants explicitly reported that the job of accountants is repetitive in general. The activities that were more frequently described as repetitive consisted in entering data in pre-formatted forms or tables or dealing with paperwork where only few details had to be changed from case to case. Some of the reported activities consisted of long sequences of tasks that were connected by temporal (e.g. you can start task B only after completing task A) or logical (e.g. if task A gives as result X, then continue with task B, otherwise do task C) dependencies. Participants reported that, although some activities were repetitive, they still required a certain level of expertise and that the experience of accountants was a central factor for having things done correctly:

> [P1: "*In general, the most part of administrative work is repetitive, but we know where we can find the information we need…*"].
>
> [P3: *"In administrative jobs all the procedures are quite repetitive […] anyway some kind of attitudes and experience is required…"*].

Five of the interviewees reported that, in the course of their work experience, they developed a set of *strategies* for speeding up repetitive tasks, making them less bor-

ing, and reducing the amount of errors that can derive from the loss of attention that sometimes is connected to repetitiveness.

> [P3: *"Everyone finds his/her ways to organize these processes by creating sets of "personal routines."]*.

> [P6: *"I created my personal strategies, procedures for speeding up the process and for avoiding errors that can occur when you are tired or in a hurry."*].

These strategies usually consisted in the way a particular activity is subdivided or managed and differed from person to person. None of the participants reported any strategies that emerged from collaboration with colleagues or from instruction provided by supervisors/heads. Participants also talked about a number of software *tools* that were developed for facilitating administrative work. Some of these tools (e.g. Word™ or Excel™ templates, checklists) were developed directly by users and appeared to be shaped on the individual need and personal preferences of the users and their practices. Other tools appeared to respond more to needs at a "office level" – as opposed to "individual level" - (e.g. databases, shared documents or calendars, web-based applications) and were developed either by accountants that had a better knowledge of computer systems or by software engineers. These strategies and tools were perceived as facilitating work because they were effective in reducing or optimizing the amount of time that activities require, maintaining organization and priorities among tasks, and reducing the number of errors.

When asked if they ever tried to develop some simple programs for automatizing activities that are made by sequences of simple tasks, only two participants responded positively. In one case (P9), the participant wrote a meta-code for an application for the management of exams, which was then implemented by a technical developer; in the second case (P4), the participant reported about a database she developed using Microsoft® Access™ and that was then re-implemented by the system operators of the Department of Computer Science with the purpose of augmenting and customizing its functionalities. In both cases results were reported as being extremely satisfactory. The most part of the interviewees, though, never tried to design software beyond the ready-to-use functionalities of the software they use, and explained this as part of their lack of software engineering skills.

Before seeing the example of automatization of the workflow implemented using Apple Automator, participants were asked how they would have performed the same sequence of tasks in the case this activity had to be performed every day. None of the participants suggested a solution that included automatization of any of the steps. However, after seeing the Automator example, all the participants reported that such software would be of great help in their everyday work:

> [P2: *"A software like this seems useful; I would spend some time to learn how to use it; having quick positive results would be a good motivation for its future use; if first results are not good I would ask help to a system operator"*.].

> [P10: *"This program would be very useful to solve problems that usually take a lot of time"*.].

When asked if they would prefer to automatize an activity using software similar to Automator or to maintain their usual manual procedure in the case of an urgent upcoming deadline, seven out of ten of the interviewees declared that they would prefer to try to automatize the process. However, they put forward some conditions that would be crucial for motivating them to choose automatization: 1) the automated procedure should guarantee to be effective, ready-to-use and safe; and 2) the support of technicians should be available during the process of automatization.

Help and Advice. Understanding how people usually look for and receive help, advice and solutions to problems that may occur while using computers at work is an important question if we want to develop new forms of computer-provided help and suggestions to users. Asking help to colleagues and system operators represented the first option for half of the interviewees. The person to whom they asked for help was usually chosen on the basis of his/her level of expertise in the specific domain in which the problem fell into. Also the level of friendship or acquaintanceship played an important role in the choice of the person. Google represented the first choice of help for four of the participants and the second choice for those participants who could not find a solution to their problems by asking colleagues or system operators. Participants reported using online help and Help menus rarely, and this was the first choice only for 1 interviewee. Advantages and Drawbacks of each of the methods reported by participants are described in Table 4.

Table 4. Advantages and drawbacks of methods for asking help reported by interviewees.

Strategy	Advantages	Drawbacks
Asking colleagues / system operators	Quick and correct answer without wasting time looking elsewhere	Colleagues and system operators are not always present. If expert people always solve problems, you never learn how to solve them by yourself.
Look up solution in Google	Offers quick solutions without bothering other people (especially for simple problems).	-
Online help / help menu	Offer more complete reference.	It is difficult to understand. It makes you waste time. Require precise queries.

When asked which one was the most effective way for receiving help, among the options they reported, eight participants indicated colleagues and technicians. Their choice was motivated by the fact that technicians are very professional and helpful, and that providing support is part of their job. One participant indicated Google as the best source of information *"because you can use it at any time, also when you are at home"* (P10). One interviewee indicated paper manuals (when not in a hurry) and the Help menu (when time is short) as the most effective methods, because *"they provide very complete information"* (P7).

When asked which method for providing help and advice they preferred between *automatic/contextual help* and *help on demand*, seven participants reported that they prefer automatic/contextual help, but two of them also specified that this method works better for new or simple applications, while for more complex or well estab-

lished procedures they would prefer help on demand. One participant provided an interesting observation about the function of automatic/contextual help:

> [P10: *"Automatic/contextual help has a double function: it appears when you need help and reminds you of potential errors; help on demand covers only the first function"*.].

Only 2 of the participants declared preferring help on demand for all their activities because automatic/contextual help can be a source of distraction and provide unnecessary hints. Participants also suggested that the automatic/help function should be customizable in order to be really useful:

> [P9: *"I prefer the automatic help because it makes me waste less time and spots out errors, but I should be able to deactivate it if I don't need it"*.].

Participants where shown some examples of contextual help and asked to comment on their effectiveness and usefulness. The function of *automatic word completion* by Google was considered very useful by all the participants. They reported that this function helped them in typing queries more quickly and without spelling errors, and that it also provided useful suggestions for other topics that are related to the desired keywords. The *Automatic friend suggestion* in Facebook was considered helpful by only five of the interviewees. However, this function was perceived as too intrusive in people's personal life and some participants declared that they preferred to manually look up for and add their new friends. Five of the participants declared to use the functions that *suggest book titles* in Amazon (i.e. "frequently bought together", or "customers who bought this item also bought") and to find it very useful for finding new books they did not heard of before. Two participants reported that they did not use this function but that they found it potentially useful; three participants reported to be annoyed by this function: they do not like the idea of using a system that interprets their taste for commercial purposes and that "*restricts, instead of expanding, the field of search*" (P4). The possibility of *automatically saving passwords* was considered useful by six of the ten interviewees but two of them also reported the concern that it can be dangerous for security of sensitive data, which is the main reason for four participants not to use it. Willingness to use such function heavily depends on the level of trust that people have on technology, especially when dealing with sensitive data. Five of the participants reported to use on a daily basis the function of Google Calendar that reminds of approaching meetings or commitments. The *related videos* sidebar featured in YouTube was considered very useful by nine of the interviewees; these suggestions were considered relevant, inspirational, and helpful for discovering new things.

When asked to formulate their "theories" about how the suggestions were generated in services like Google, Amazon, Facebook and YouTube, all the participants reported that they are created on the basis of the inserted keywords; one participant also made a distinction between general, or simple, and particular, or complex, suggestions:

> [P8: *"For simple queries, the system works on simple analogies with the inserted keywords; for more complex issues, the system*

does a matching with your personal characteristics (provided while registering to a service)".].

When asked if these suggestions are effective in meeting personal taste and needs, five participants responded positively but two of them also specified that these systems work well only in the case of "objective" suggestions, where the system does not make hypotheses about the user's personality (e.g. in YouTube). Suggestions are considered to be less accurate when they try to enter users' private space (e.g. in Facebook):

> [P4: *"The suggestions are good when referring to "objects"; when they try to catch my personality, they are not effective"*.].
>
> [P10: *"Suggestions work well for simple things (Amazon, YouTube), but when they try to get more personal, then they are less accurate"*.].

WIRE. Participants provided very useful information about their attitude toward WIRE. When reading the positive scenario, participants recognized several similarities with their work practices and perceived the system as potentially very useful:

> [P2: *"The accountant described in the scenario is very enthusiastic, I also would be like that if I had the right instruments and a good training; WIRE helps in keeping a structure of the work"*.].
>
> [P10: *"Awesome! This is exactly what a system should do to help people"*.].

Two participants expressed a common concern about the introduction of such system into their common work practices and suggested that, in order to benefit of its potentialities, the use of WIRE should totally replace previous practices, without leaving space for overlapping of the two methodologies:

> [P3: *"I am not sure about the way the accountant shares his experience, few colleagues do that, sharing should be forced from above"*.].
>
> [P10: *"One weakness could be the difficulty in recreating a process ex-post (after it actually occurred): software are less flexible than people; a software like this should force people to use it: if people are given the chance to use alternative ways in parallel, then things can become complicated"*.].

The Negative Scenario was also perceived as very plausible as it described well fears and frustrations that may emerge when something goes wrong while using computers, especially when dealing with new systems or procedures. Some interesting observations were provided by interviewees about the importance of a system that is well designed and thoroughly tested before being introduced into the work practice:

> [P7: *"I gave for granted that this technology was previously tested and approved by the central administration office. If I don't entirely trust it, I would prefer to perform some manual cross-checks. In the*

case of dealing with sensitive or financial matters, I would trust the system only if I am 100% sure that it is effective and functional".].

[P9: *"This scenario shows a case in which something went wrong on the technical side; things go wrong when technology is not well designed"*.].

Participants were asked if they would be interested in using WIRE. Nine of the interviewees responded very positively and one was openly sceptical about the new system, adding that *"using WIRE would take the same time it takes doing the procedure manually"* (P1). Anyway, a period of formal training was indicated by two participants as a fundamental prerequisite for the adoption of WIRE:

[P3: *"I would use it only after receiving a very good training and if it is not too complicated"*.].

[P10: *"It is very important to invest some time to learn new instruments"*.].

Participants indicated *better organization of work*, *optimization of time*, *reduction of errors*, and *sharing of procedures and methodologies with colleagues* as the most relevant benefits connected with the use of WIRE. The risk of a *loss of control over work processes*, in the case that these were entirely completed in an automatic way, was indicated as the major potential drawback of the system:

[P4: *"I would like to keep track of each step of the process; if everything is made automatically, the users misses the logic that stays behind the process"*.].

[P6: "*If the procedure is too automatic, if I do some mistake in inserting values, then I will not be able to correct my mistakes. Users must be always aware of what they are doing, with a system that is too automatic, some people could get distracted"*.].

All the participants reported that the kind of help and suggestion that WIRE provided in the scenarios would be effective in meeting their personal needs, as the contents of the help message was created on the basis of past experience of colleagues that share the same work procedures and possibly the same difficulties:

[P6: *"I think that suggestions are useful because they are based on well established and shared experiences"*.].

[P9: *"If suggestions come from people of the same area, who share the same problems I have, then I think that these suggestions are valid"*.].

All the interviewees reported that the help that they would be effective in supporting their work; two of them added that the possibility of personalizing the way suggestions are provided would be a very important feature in order to make help messages really effective. Eight out of the ten participants reported preferring automatic/contextual help to help on demand in the case of an application like WIRE. Two participants motivated this choice by adding that automatic help can remind users about steps that they maybe would forget.

Help messages provided *during the task* were preferred to messages provided before the task by nine of the interviewees. One participant suggested that the two modalities could be combined:

> [P8: *"I can see the two modalities as complementary. At the beginning of the activity the system asks what your needs are in general; during the activity, pop-up windows provide you solutions when the system feels that you are stuck"*.].

5 Conclusion

Consistently with [1] and [2], our study showed that people who do not have a specific background in computer science or software engineering know very little about SOA and web-services. They tend to define web-services following the mental model which they have consolidated in real life, as that of a third party which does something on their behalf. In this view, web services are perceived as something that provides assistance to perform actions on-line: they can be any type of software, from online banking to search engines, or Google documents. Participants were not aware of any technical detail about the meaning that the word has assumed in software engineering, nor of the possibility of using services to build or personalize their applications. These results challenge the emerging idea of SOA as a simple panacea to open software engineering to end-users, and call for more research into innovative metaphors to make SOA concepts and tools available to a larger population of non-technical users.

The interviews highlighted the fact that software are still perceived as tools to be used "as they are", and that customization of their functionalities is an option that usually accountants do not consider. The main barriers to EUD uptake were identified as 'concerns over reliability and security of the resulting artifact', 'need for control', 'need for help', and 'organizational barriers'. A clear tension between the requirements for simplicity and user control emerged. If on the one hand, the fact that building blocks were available to be easily combined by the users was considered an important advance over their previous experience with programming tasks, people declared to be reluctant to set-up fully automated systems to support their work, because of their apprehension of loosing control over the intermediate steps. Component-based programming was perceived as risky, because the user did not have direct control over the procedures implemented in each component. These results suggest the importance of developing a transparent system in which the users can keep a high level of control and monitoring over the processes they are dealing with.

End-users acknowledged that the idea of WIRE of providing assistance derived from the experience of colleagues working in a similar context had potential. However, issues of trust, timing and usefulness of the advice still remained important. During the design of WIRE we will need to find new strategies to make transparent how the advice was generated in the form of trust seal, particularly important when people deals with sensitive and financial issues. *Personalization* of procedures is also a desired feature: the system should be able to take into account the strategies that people used to optimize their work practices during the past. Giving users the func-

tionality that their expertise is taken into consideration while developing new practice is likely to facilitate the adoption by end users of a new system.
Finally, our data provide support to the proposal of collaborative tailoring, discussed in [12], as often participants mentioned that their willingness to engage in EUD was mediated by having support from other people and technical help easily available to them. This help was meant not only to alleviate some of the technical difficulties they had to face during development but also to take the responsibility out of their hands, making them less accountable in case of software failures. Issues related to organizational regulations and corporate processes also emerged as barriers to EUD uptake, as people often mentioned the need to have unambiguous approval from their manager as a fundamental step towards making them willing to explore new techniques and tools to automatise their work practices.

Acknowledgments. This work was partially supported by funds from the European Commission (project OMELETTE, contract no. 257635; project SERVFACE, contract no. 216699).

References

1. Namoun, A., Nestler, T., De Angeli, A.: Conceptual and Usability Issues in the Composable Web of Software Services. In: Daniel, F. and Facca, F.M. (eds.) Current Trends in Web Engineering - 10th International Conference on Web Engineering ICWE 2010 Workshops, Revised Selected Papers. LNCS, vol. 6385, pp. 396-407, Springer, Heidelberg (2010)
2. Namoun A., Nestler T., De Angeli, A.: Service Composition for Non Programmers: Prospects, Problems, and Design Recommendations. In: 8th IEEE European Conference on Web Services ECOWS (2010)
3. Cypher, I., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., Turransky, A. (eds.): Watch what I do: Programming by Demonstration. MIT Press, Cambridge, MA, USA (1993)
2. Van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases. 14(3), 5--51 (July2003)
3. Henneberger, M., Heinrich, B., Lautenbacher, F., Bauer, B.: Semantic-based Planning Process Models. In: MKWI'08, Munich, Germany (2008)
4. Ngu, A., Carlson, M., Sheng, Q., Paik, H.: Semantic-Based Mashup of Composite Applications. IEEE Transactions on Services Computing. 3(1), 2--15 (2010)
5. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for Mashups. In: Proc. VLDB Endow. 2, no. 1, pp. 538-549 (2009)
6. Yue, K.: Experience on mashup development with end user programming environment. Journal of Information Systems Education. 21(1), 111--119 (2010)
7. Roy Chowdhury, S., Rodríguez, C., Daniel, F., Casati, F.: Wisdom-Aware Computing: On the Interactive Recommendation of Composition Knowledge. In: Proceedings of WESOA 2010, Springer (2010)
8. Daniel, F., Casati, F., Benatallah, B., Shan, M.-C.: Hosted Universal Composition: Models, Languages and Infrastructure. In mashArt. ER'09, pp. 428-443, (2009)
11. Bødker, S.: Scenarios in user-centred design - setting the stage for reflection and action. Interacting with Computers. 13, 61--75 (2000)

12. Wulf, V., Pipek, V., and Won, M.: Component-based tailorability: enabling highly flexible software applications, International Journal of Human-Computer Studies. 66(1), 1--22 (2008)

Appendix: The Scenarios

Introduction (common to both scenarios). Alberto is an accountant at the University of Trento: one of his duties is the reimbursement of travel expenses. This is a sensitive and time-consuming work which requires several repetitive tasks (e.g., checking staff profile, projects details, financial reports and regulations, obtaining proper authorization), which Alberto performs manually. He was told that the university recently bought a web-tool called *WIRE,* which can help him to make repetitive tasks automatic. Although he has never done any type of development before, he is curious to try. A short on-line video shows him that *WIRE* works by connecting *activities*. WIRE provides a set of activities similar to the paperwork Alberto is familiar with; a composition area where these activities can be assembled; and a set of contextual advices generated according to the actions of the user.

Positive Scenario. Alberto starts designing an application for the reimbursement process. He knows from his daily experience that the process starts with a request, so he searches for this activity. He finds an activity called *Reimbursement request* and, after reading its description, he places it in the composition area. Then, he selects the *Get trip document* activity and at that point an automatic advice suggests him to take also the *Translator and* the *Currency Converter* activities. Alberto knows that the advice is right because he deals with many travels in foreigner countries. He thus accepts the advice and WIRE automatically places the two activities on the composition area adding the necessary connections to the other activities. Alberto progresses the design choosing all activities needed and WIRE help him suggesting how to connect them. Once Alberto thinks he has finished, WIRE proposes to add the activity C*heck Authorization* in a specific point of the workflow. He remembers that this task is mandatory and needs to be performed early, so he gladly accepts the advice and the system automatically update the configuration. Now Alberto decides to share the application with all his colleagues: he clicks on the link Publish and the application is automatically posted on the Intranet.

Negative Scenario. The first impression that Alberto has is mixed. WIRE provides a large number of activities that certainly will include those he needs but it is difficult to understand what they do and how to use them. Alberto fears that learning will take quite some time. However, he decides to give it a try and starts designing an application. He knows from his daily experience that the process starts with a request, so he searches for this activity. He spends some time to find an activity called *Reimbursement request* and he places it in the composition area. Then, he selects the *Get trip document* activity and at that point an automatic advice suggests him to take also *Translator and Currency Converter*. Alberto does not need these activities as the application he wants to build is only for national trip. He finds the advice annoying, as he knows exactly which activities are involved in his job and was distracted by the unnecessary suggestion, which took time to be understood. Alberto progresses the design choosing all activities he needs but then he feels unsure on how to connect them. The system however does not have any advice on the problem. Alberto is not sure about the correctness of the application he created. He is worried about the fact that it will handle money and sensitive information of which he is responsible for. He misses the level of control that he feels while processing his task manually one after the other so he goes back to the traditional work.