Implementation of End-User Development Success Factors in Mashup Development Environments

David Lizcano[1], Genoveva López[2], Javier Soriano[2], Jaime Lloret[3]

[1]Universidad a Distancia de Madrid, Camino de la Fonda 20, 28400, Collado Villalba, Madrid, Spain

[2]Universidad Politécnica de Madrid, Campus de Montegancedo, 28660, Boadilla del Monte, Madrid, Spain

[3]Universidad Politécnica de Valencia, Camino Vera s/n, 46022, Valencia, Spain

Abstract

The Future Internet is expected to be composed of a mesh of interoperable web services accessed from all over the Web. This approach has been supported by many software providers who have provided a wide range of mashup tools for creating composite applications based on components prepared by the respective provider. These tools aim to achieve the end-user development (EUD) of rich internet applications (RIA); however, most, having failed to meet the needs of end users without programming knowledge, have been unsuccessful. Thus, many studies have investigated success factors in order to propose scales of success factor objectives and assess the adequacy of mashup tools for their purpose. After reviewing much of the available literature, this paper proposes a new success factor scale based on human factors, human-computer interaction (HCI) factors and the specialization-functionality relationship. It brings together all these factors, offering a general conception of EUD success factors. The proposed scale was applied in an empirical study on current EUD tools, which found that today's EUD tools have many shortcomings. In order to achieve an acceptable success rate among end users, we then designed a mashup tool architecture, called FAST-Wirecloud, built taking into account the proposed EUD success factor scale. The results of a new empirical study carried out using this tool have demonstrated that users are better able to successfully develop their composite applications and that FAST-Wirecloud has scored higher than all the other tools under study on all scales of measurement, and particularly on the scale proposed in this paper.

Keywords: End-user development, mashup tool, service front-ends, service-oriented architectures, HCI success factors

1. Introduction

Service-oriented architectures (SOA) have attracted a great deal of interest over the last few years. In fact, SOAs increase asset reuse, reduce integration expenses and improve business agility in responding to new demands [1]. Nonetheless, mainstream development and research into SOAs have until now focused mainly on middleware and scalability, service engineering and automating service composition, using business modelling process technologies. Little or no attention has been paid to service front-ends, which are a fundamental part of SOAs [2].

As a result, SOAs remain on a technical layer hidden away from the service end user, called end-user programmer (EUP), a person who programs to achieve the result of a program primarily for personal rather than public use [35]. The evolution of web-based interfaces bears testimony to the progress made towards improving service usability. However, existing web-based service front-ends do not come anywhere near to meeting EUPs' expectations [3]. Applications and EUP tools are still based on monolithic, inflexible, non-context-aware, non-customizable and unfriendly user interfaces [4]. Consequently, EUPs do not really benefit from the advantages promoted by service orientation in terms of modularity, flexibility and composition [5].

The Web 2.0 social and technological movement highlighted the need to involve service-based consumer portals, web content and web applications in the development, adaptation and improvement of such applications. Accordingly, data mashups and interfaces, and mashup development environments, have come to prominence in recent years. Large companies have earmarked part of their investment for providing service front-ends for applications and data on which they base their business value. These front-ends are adapted to and adapted by the user in order to lower the barrier between the technology layer of a SOA-based application and the EUP [6]. This DIY (do-it-yourself) approach has the backing of large companies like Google (originally via iGoogle and then Chrome Web Store), Yahoo! (with Yahoo! Pipes and Yahoo! Dapper), Microsoft (with PopFly) or IBM (with SOA4People and QEDWiki) among others [7].

Their aim is to get EUPs to appreciate the benefits of SOA by fostering composition, loose coupling and reuse on the front-end layer, and moving towards a user-centred service conception [8]. Thus the above web development tools aim to empower EUPs to create their own mashups of data and service interfaces, which are organized so as to cover their basic needs. The resulting expenditure of time and effort should be well below that of traditional compositional development, which is based on integrating and organizing back-end services and resources and requires advanced programming skills that an end user does not have.

There is, however, an epistemological problem: although they target EUPs, resources and mashup tools are not enough to ensure that EUPs can develop their own solution to a particular problem, primarily because they have not been taught how go about this. In most cases, EUPs do not perceive themselves as being able to translate requirements to a mashup that meets such requirements.

This is less of a problem if the components of these mashups and the employed compositional techniques tie in with the end-user cognitive model [9]. This has been achieved with great success in the spreadsheet field, where end users can create a spreadsheet application by establishing data flows between cells and preconceived functions that they neither have to be acquainted with nor have to program. Our research is based on applying the success factors of this domain, commonly called end-user success factors, to the mashup field, to provide EUPs with a web development dataflow model between heterogeneous and dispersed service interfaces. To do this, we have to consider which actions, principles and objectives, and design decisions can be derived from replicable success factors in the spreadsheet domain and study their applicability to web composition tools targeting EUPs.

In this paper we will review the most relevant publications and scientific results on end-user development (EUD) success factors, which we will then combine in an innovative architecture to study EUD success factors in the web domain. Thus, we will be able to study each web mashup tool and form a general idea of how successful it is likely to be among EUPs based on the studied factors. Additionally, we will demonstrate the usefulness of this study for guiding key design and architectural decisions in order to achieve EUP satisfaction, which is vital for improving future EUD solutions. For this purpose, we will present the architecture of a web mashup tool that we have developed and highlight how this architecture performs on the proposed scale for studying EUD success factors. The remainder of the paper is structured as follows. Section 2 presents the state of the art and related work on end-user development tools with respect to well-known and commonly accepted human and human–computer interaction (HCI) success factors and successful specialization/functionality trade-offs. The proposed scale for studying end-user development success factors is presented in Section 3. Section 4 shows a performance comparison of our proposal according to traditional measurement scale success factors. On the basis of the proposed measurement scale, Section 5 proposes a novel architecture for end user-centred service front-end development that is based on the proposed measurement scale. We have used this architecture to drive the development of an improved mashup tool that is presented in Section 6. Sections 5 and 6 both illustrate how this mashup tool architecture performs with respect to the reported EUD success factors and that it offers an appropriate specialization/functionality trade-off. Section 7 evaluates this mashup tool using the end-user development success factor scale. Section 8 discusses threats to validity. Finally, Section 9 outlines the conclusions.

2 Related Work

Nowadays there are several applications empowering EUPs to develop their own software solutions, which are adapted to their unique and instant requirements. These applications, like spreadsheets, e-mail filter creators or mashup web tools, focus on outputting different software solutions, each oriented to a specific problem domain, such as calculation requirements, spam filtering or visual web widget composition. Of all these approaches, several studies state that spreadsheets are the major and most successful EUD solution existing at present [10]. In an empirical study carried out by Wu et al. [10], 100% of a huge sample of EUPs had at some time used a spreadsheet program in their daily work to solve one problem or another. Other publications, like Boehm [11], suggested that more than 55 million people in the United States do this kind of spreadsheet programming, whereas professional programmers account for about 2.75 million of the country's population. This gap is actually widening, as Scaffidi [12] predicted that the EUD population (users of spreadsheets and other EUD approaches) at workplaces in the United States would be 100 million by 2015. These studies and publications indicate that EUD is about to take control not only of personalizing computer applications and writing programs but also of designing new computer-based applications without ever seeing the underlying program code.

For this reason, many researchers have begun to study EUD success factors [13], focusing above all on the most relevant EUD solution, spreadsheets, to understand which of their principles and factors are used and accepted by EUPs. Studies like [10] focus on successful human EUD factors, whereas other studies like [14] focus on HCI factors or on aspects of specialization and functionality [15]. The feedback that we got after reviewing all referenced studies is that EUD success is related to human factors, HCI factors and the specialization-functionality relationship. However, there are no publications that put all these ideas together to offer a general conception of EUD success factors. Besides, we have not found any reliable reference to other success factors that could lead an EUP tool to successfully achieve such objectives. In this paper, we propose a joint scale that brings together all the success factors considered in previous studies by other researchers and ourselves, which measures the extent to which an EUP tool has such factors and what impact they have on the achieved results and users.

2.1 Human Success Factors

All software development tools should be well accepted by their target users if they are to be considered a successful solution. However, this acceptance, as Wu et al. [10] show in their empirical survey, is not down to the choice of particular technologies or architectural decisions, but to whether or not they preserve and take care of a number of human factors.

End-user computer acceptance has been established by Wu et al. [10] as one of the critical success factors in achieving business success and is defined as the adoption and use of information technology by personnel outside the IT domain to develop software applications in support of organizational tasks. Davis [16] proposed the technology acceptance model derived from reasoned action theory that has been tested and extended by numerous empirical researchers. In these studies the actual use of any application is derived from several human factors like perceived ease of use, perceived usefulness, and so on, and these ideas were the basis for Wu et al.'s research. The empirical study carried out by Wu et al. [10] relied on 800 people testing programs and evaluating software solutions. The evaluation showed that actual software use follows the causal relationships illustrated in Figure 1. This diagram establishes which factors are related to actual end use, and what weight or strength this causal relation has, expressed by a correlation coefficient between two factors. The correlation coefficient, ranging from -1 to 1, indicates that the correlation between the factors is positive and stronger the closer it comes to 1. A negative correlation, which is stronger the closer it comes to -1, would indicate that the value of one factor decreases, as the value of the other increases. If the value is 0, there is no correlation between the factors. The key factors are explained in detail in Table 1.

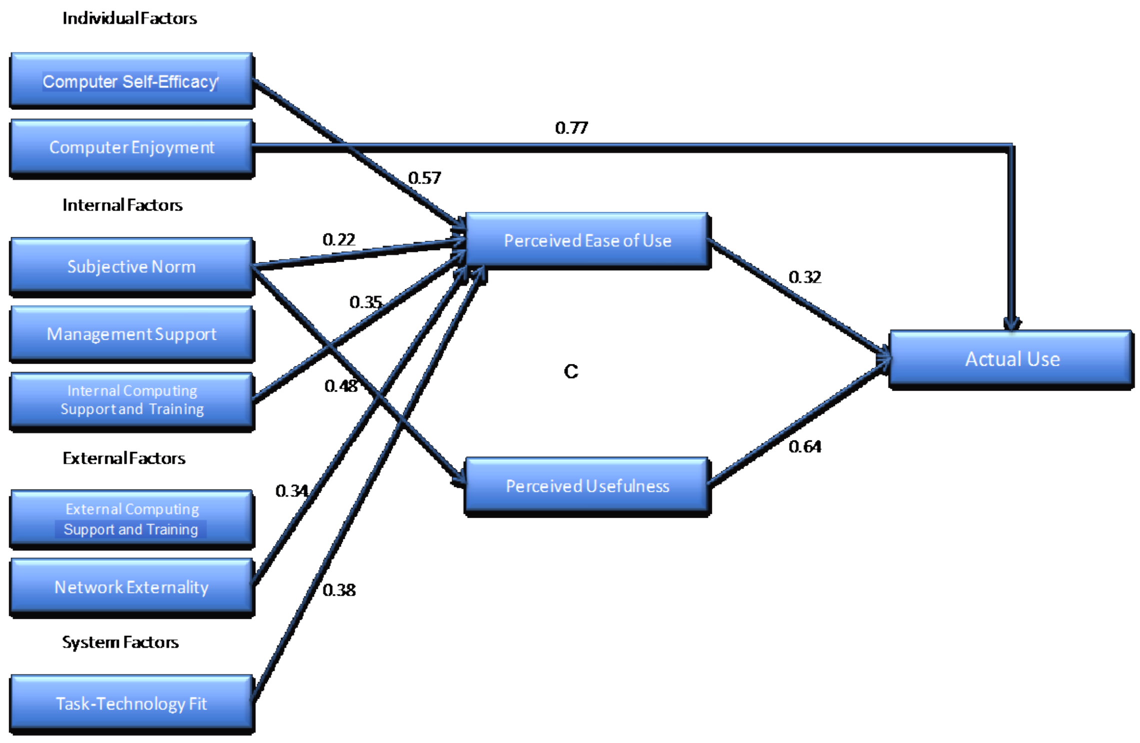


Figure 1. Empirical results of study about human factors related to EUD success.

Computer Self-Efficacy	A person's perception of his or her ability to use computers in the accomplishment of a task.

Computer Enjoyment	Individuals experience immediate pleasure and joy from using software
Subjective Norm	The degree to which a person believes that people that are important to him or her think that he or she should do the thing in question
Management Support	Perceived level of general support offered by top management
Internal Computing Support and Training	Technical support and the amount of training provided within the company.
External Computing Support and Training	Technical support and the amount of training provided by individuals from outside the company.
Network Externality	The utility of software use increases if its number of users increases.
Task-Technology Fit	The degree to which an organization's applications meet the information needs of the task.
Perceived Ease of Use	The degree to which a person believes that using specific software would be free of effort.
Perceived Usefulness	The degree to which a person believes that using specific software will increase his or her job performance.

Table 1. Human factors related to EUD success (actual use).

Figure 1 includes factors with multiple weighted paths to the final concept of actual use, and therefore the correlation coefficient between each factor and the use of a program is not clear. All correlation coefficients need to be recalculated for each new application domain in order to reveal the final impact of each factor on the actual use of the software. Figure 2 shows the coefficients originally calculated by Davis [16] and later refined and applied by Wu et al. [10] to EUD success. It is thus possible to calculate the final impact of each factor on the actual use of service-oriented software. Accordingly, each factor can be scaled to give an idea of its importance.

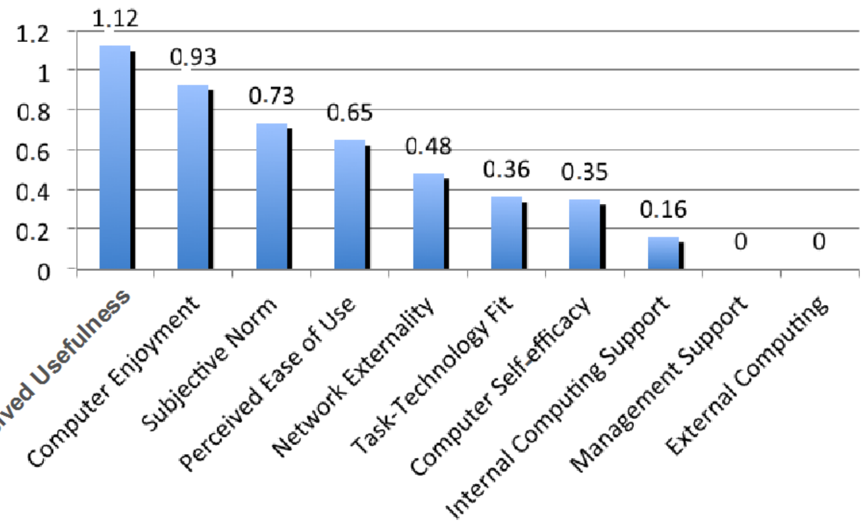


Figure 2. Relevance of each human factor related to EUD success in web software.

The above research suggests that an end user will use a program if he or she perceives it to be useful and enjoys the experience of using it. If other people in the end user's environment use the application, the end user is more likely to accept and use this software, too. Finally, ease of use and the fact that application usefulness would increase if it were used by more and more users will lead to more actual use of a software tool.

2.2 HCI Success Factors

Other studies like the one presented by Jones et al. in [14] claim that spreadsheets (and other similar EUD solutions, such as web mashups and visual end-user programming IDEs) are the programming language of choice for many people because of their HCI facilities. Spreadsheets are a user-centred approach to language design, focusing on fostering usability through effective HCI. Specialized research into the psychology of programming and empirical studies of programmers [14] offer groundwork for human issues in programming, structured as cognitive dimensions, that EUD solutions should consider and optimize in order to be successful among EUPs [13]. These cognitive dimensions are in fact HCI factors that, if properly taken care of, result in high end-user acceptance on a par with spreadsheets [17]. Green and Petre [18] defined 13 cognitive dimensions (nine of which are of equal importance, the other four being minor dimensions more related to user-related cultural issues) that if properly addressed, improve HCI and simplify EUD. The most often cited factors are listed below:

- Abstraction gradient: What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?
- Consistency: When some of the techniques, methodology and programming language have been learnt, how much of the rest can be inferred?
- Error-proneness: Does the design of the notation induce “careless mistakes”?
- Hidden dependencies: Is every dependency overtly indicated in both directions? Is the indication perceptual or merely symbolic?
- Premature commitment: Do programmers have to make decisions before they have the information they need?
- Progressive evaluation: Can a partially complete program be executed to gather feedback on “How am I doing”?
- Role expressiveness: Can the reader see how each component of a program relates to the whole?
- Viscosity: How much effort is required to make a single change?
- Visibility and juxtaposability: Is every part of the code/development simultaneously visible, or is it at least possible to compare any two parts side by side at will? If the code is dispersed, is it clear in which order it should be read at least?

According to Jones et al.’s study [14], software that accounts for these factors, like Microsoft Excel or other spreadsheet and end-user programming solutions, enable EUPs to implement software in a simple and flexible manner. Therefore, these dimensions must be kept in mind when new EUD approaches are set out.

2.3 Successful Specialization/Functionality Trade-off

For many researchers in the EUD and composite applications domain, the key factor for success among EUPs is that EUD software accomplishes a good trade-off between the specialization and the functionality of the resulting solutions [15]. Some mashup approaches recently started to work on "domain specificity", i.e. the tailorability of a mashup tool to specific requirements that may arise in specific domains (e.g. in specific working communities) [44]. However, solutions focusing on a single domain are unable to address complex problems that require solutions involving heterogeneous interdisciplinary components, that is, they do not have the functionality to solve generic problems. There is, therefore, a need for a trade-off between two opposite approaches: the EUP tool should be able to build specialized solutions in a domain and have sufficient functionality to be able to do the same in domains that are quite unalike. This trade-off gives an idea of whether EUPs would be able to build their own solutions to satisfy their needs [19].

How well suited a developed solution is for a task or real problem could be quantified by two factors:

- Specialization: the degree to which an application exactly matches real requirements, features and details of a real problem, without the need for the user to have to further adapt it to the problem domain.
- Functionality: the sum or any aspect of what a product, such as a software application or computing device, can do for a user. The overall functionality decreases when the solution is overly specialized for a specific problem.

As these are opposing factors, however, it is impossible to increase one factor without decreasing the other. For this reason, EUD solutions should adopt a trade-off where both factors are at equilibrium [19]. This will lead to solutions that are very specialized for a problem but could easily be exported and used in other problem domains. This balance is frequently measured on a four-point Likert scale (poor, average, good or optimal specialization/functionality relationship) [15].

In this paper we detail how we built a new measurement scale for EUD success factors which is based on a previous proposal [20]. We also statistically validate the scale by demonstrating that all the families of factors used on separate axes are independent. Additionally, we report a case study illustrating performance on this new scale, which found that the existing tools ranking top on the proposed scale are the ones using which EUPs are more efficient. Accordingly, if current mashup tools were to score higher on the proposed scale, they would be more successful among EUPs and more usable. This would make current tools and environments better at supporting EUD processes.

3. Proposal

All the factors studied in the literature are frequently referenced and used in EUD research, but they are always applied individually. In this paper, we propose the creation of a complex scale to study key EUD success factors globally by combining all the studied factors. Since each factor type targets one axis of the scale structure, we have concluded that, as demonstrated in [42], human factors, HCI factors and the specialization/functionality factor are orthogonal to each other, that is, they are mutually independent, uncorrelated factors. Considering this premise, which is validated in the next section, each family of factors can be integrated as an independent axis on a complex 3D scale that represents each independent family of factors (Figure 3). Each axis must be managed as follows:

- X-axis = human factors. In the last section we described eight factors that should be considered to achieve EUD success. These factors had correlation coefficients denoting their relevance. On the proposed scale, the study of an EUD solution will include an evaluation of each factor on a three-point Likert scale (0 for a low factor rating, 1 for an average rating and 2 for a high rating). This rating will be multiplied by the correlation coefficient (shown in Figure 2) to output a final rating for this factor. Each factor must be evaluated and added together for each EUD solution studied. This will add up to a final rating of from 0 to 9.56 (as a result of the correlation coefficients). Finally, this rating has to be normalized to a standard scale ranging from 0 to 10 (by multiplying by 10/9.56). This final value will be represented on the X-axis to illustrate how successful the studied solution would be in terms of EUD based on human factors.

- Y-axis = HCI factors. In the previous section we studied thirteen HCI factors that should be improved to achieve EUD success, all of which were equally important. Therefore, an analysis of these factors for a EUD solution will involve evaluating each factor on a three-point Likert scale (0 for a low rating, 1 for an average rating and 2 for a high rating of the factor) and adding up this value for each factor. This process will output an overall rating ranging from 0 to 26 points. Finally, this value has to be normalized to a standard scale (0-10) by multiplying by 10/26. This final value will be represented on the Y-axis to illustrate how successful the studied solution would be in terms of EUD based on its cognitive dimensions.

- Z-axis = Specialization/functionality trade-off: the four-point Likert score studied in the related work is represented directly on the z-axis, adding a third dimension to the expected EUD success of a solution.

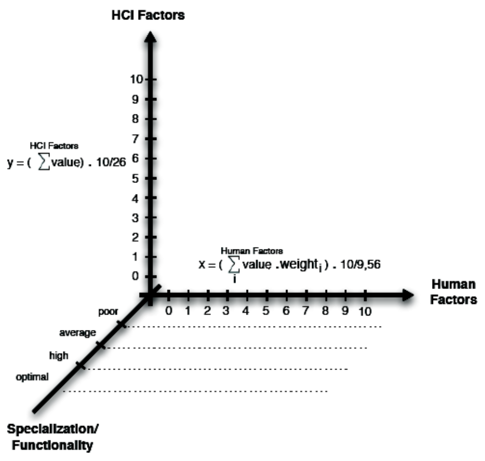

HCI Factors
10
9
8
7
6
5
4
3
2
1
0
HCI Factors
$y = (\sum \text{value}) \cdot 10/26$
Human Factors
$x = (\sum_i \text{value} \cdot \text{weight}_i) \cdot 10/9{,}56$
0 1 2 3 4 5 6 7 8 9 10
Human Factors
poor
average
high
optimal
Specialization/ Functionality

Figure 3. Proposed scale for studying EUD success factors.

We believe that this theoretical framework for the study of success factors is very useful in two ways: It is a powerful scale for rating any EUD tool for EUP and for forming an idea of expected EUD success based on widespread and proven principles, founded on factors included in several referenced research papers, and it also summarizes all proven factors that are related to actual use and user acceptance. It is therefore a good starting point for creating new EUD environments or approaches.

4. Performance comparison

To verify the effectiveness of each measurement scale success factor, we conducted a study in response to the following research questions:

- RQ1: Are there correlations between the different scales of measurement considered, or are they independent of each other?

- RQ2: Which measurement scale (human factor measures, HCI measures, specialization / functionality measures or a combined measurement, as proposed in this paper) is better at rating the success of a given EUD tool?

To address these two issues, we ran an experiment with 100 EUPs, characterized according to Table 2.

		Sample division into work groups			
Characterization	**EUPs (100)**	**Yahoo! Dapper and Pipes group**	**Open Kapow group**	**Chrome PT group**	**Popfly group**
Gender					
Male	51	13	13	12	13
Female	49	12	12	13	12
Age					
< 20 years	19	5	4	5	5
20-34 years	23	5	6	6	6
35-49 years	22	6	5	5	6
50-64 years	21	5	6	5	5
> 65 years	15	4	4	4	3
Educational Attainment					
Secondary School	23	5	6	6	6
Vocational Training	27	7	6	7	7
Bachelor's Degree	25	6	7	6	6
Master's Degree	25	7	6	6	6
Employment					
Student	28	7	7	7	7
Researcher	30	7	8	8	7
Employee	42	11	10	10	11
Experience and previous knowledge					
Mashup Tools	3	0	1	1	1
Web Services (SOAP, ESB, BPEL, etc.)	2	1	0	0	1
HTML, CSS	1	1	0	0	0
Java, J2EE	0	0	0	0	0
JavaScript, AJAX	0	0	0	0	0
PHP, ASP	1	0	0	0	1
OO Programming	0	0	0	0	0
C, C++, C#	0	0	0	0	0
Scripting, Perl	0	0	0	0	0
Haskell, Prolog	0	0	0	0	0

Table 2. Characterization of the 100 users in the study.

These 100 users were recruited in an experiment conducted during the Computer Sciences Applied to Business Week. We were able to recruit a very broad sample (over 160 people), select a non-biased subsample, and conduct the experiment as part of a one-day hands-on seminar.

None of the users, except for three users with knowledge of mash-up tools (in this case iGoogle), two with knowledge of web services, one with knowledge of HTML and CSS, and another acquainted with PHP, ASP, have programming skills. A key step in the study is to validate the sample, statistically proving that there is no bias in the characterization of the users. On this ground, we recruited users of different ages, with diverse professional and academic backgrounds, and tried to balance each of the qualitative and quantitative variables that characterize the sample.

The sample of users was randomly divided into four equal groups to assure that none of the groups were biased by age, sex, training, employment and previous experience. The distribution is shown in the respective columns of Table 2. Each group attended a hands-on workshop with a different tool: Yahoo! Dapper and Pipes, OpenKapow, Chrome PT (productivity tools) and PopFly. An ANCOVA study [43] was conducted to study whether the random distribution had caused bias in the variables studied in RQ1 and RQ2. The results of this study were satisfactory, specifying that the four groups were not biased and there was no correlation between the distribution and the study. The explained variable used was the group to which each user was assigned, and all the surveyed user characteristics were used as the explanatory variables. The R^2 and adjusted R^2 of the resulting ANCOVA model were more or less equal to 0 (0.0156 and 0.0014, respectively), that is, there is no way that the group to which each end user belongs is correlated to their descriptive characteristics. This means that, irrespective of the impact of each characteristic (sex, age, training, employment, previous experience) has on the study of the RQs addressed in this paper, all the analysed tools will have been measured by equivalent subsamples of users, that is, any bias is ruled out and all tools have been measured by the same yardstick.

To answer RQ1, what scale of measurement is better for rating the success of a given EUD tool, EUPs received initial training in the EUD field and in the above tools, with the following schedule of activities:

- Fundamentals session (four hours): Introduction to EUD and acquaintance with web composite applications, widgets and mashups.
- Practical session (four hours): Development of solutions on the respective platform, either Yahoo! Dapper and Pipes, OpenKapow, Chrome PT (productivity Tools) or PopFly.
- Tool survey (two hours): Completion of three surveys by users to rate their tool on each of the three scales: human factors, HCI factors, factors of specialization / functionality.
- Case study (unspecified): Hands-on experience in the laboratory developing the proposed composite application using the respective tool.

Note that the tools were rated after a practical session that we led.

The problem to be solved during the proposed case study was:

“Part of a user's routine work is to supervise changes and notifications published by two of his country's public administration webpages. This user wants to build an application that is capable of automatically reviewing these two webpages and emailing and SMS messaging these specific changes to him. Therefore he is looking for an application that does the following:

1. Stores a baseline containing the original status of the two webpages.
2. Examines the respective web pages, which do not publish RSS or notify content changes in any other way, at a user-configurable time interval and check whether any changes have been made compared against the baseline.
3. SMS message or email an outline of any changes made to the user's mobile phone and email address. Both data should obviously be configurable at runtime.”

This task requires the use of 20 integrated components, which results in a set of 100 activities (component selection, parameterization, interconnection with the prototype at development time, unit testing and integration testing). Very few users completed all 100 activities, on which ground we measured the percentage of activities completed by the 25 users that used each tool.

Table 3 shows the results for each tool, based on the score achieved by the 25 users of each tool on each of the traditional measurement scales (listing the average and median score on each scale), and the respective success rate. The questions and weights described in Figure 1 were used for human factors, producing a measure of between 0 and 9.56. The 13 factors listed in the related work section were used as HCI factors, where each factor was evaluated on a Likert scale. This process will output an overall rating ranging from 0 to 26 points. We used a four-point Likert scale to measure specialization/functionality factors: poor, average, high and optimal, ranging from 0 to 3 points. Finally, the proposed scale uses the three-dimensional scheme described in Section 3, and orthogonally sums values for each axis by weighting the result on a scale of 0 to 10, making a linear change of basis from 0-27.86 to 0-10.

	Human Factors	**HCI Factors**	**Specialization / Functionality Factors**	**Proposed Scale (Combined Factors)**	**Completed Activities**
Yahoo!Pipes and Yahoo! Dapper	Mean 4.65 Median 4.03	Mean 13.54 Median 12	Average	Mean 5.15 Median 4.56	52%
OpenKapow	Mean 5.68 Median 3.22	Mean 13.69 Median 13	High	Mean 5.37 Median 4.86	55.6%
Chrome PT	Mean 3.81 Median 4.16	Mean 13.05 Median 12.75	Poor	Mean 4.88 Median 4.81	51.2%
PopFly	Mean 2.82 Median 2.53	Mean 13.97 Median 13.47	Poor	Mean 5.11 Median 4.92	53.8%

Table 3. Results for each tool on traditional measurement scales and the number of solutions built (for all results, see [43]).

In response to RQ1, whether the different measurement scales are correlated or independent of each other, we studied the results and found that the measurements on all three scales are not dependent on each other. This was the hypothesis on which our proposal was based. To do this, we conducted a correlation and covariance analysis. Table 4 reports the final results for the different measurements.

Covariance	**Values**
Human Factors, HCI Factors	0.03127
Human Factors, S/ F Factors	0.02217
HCI Factors, S/F Factors	0.02271
Human Factors, HCI Factors, S/F Factors	0.00185

Table 4. Covariance values.

In this analysis, we did not use the measurements that we propose, because they are derived by the orthogonal summation of the values of the other scales, and are, understandably, highly correlated. As all the values are less than 0.05, we can say that, for α = 0.05, the hypothesis that human, HCI and specialization / functionality factors are independent is true.

In response to RQ2, which scale of measurement is better for rating the success of a given EUD tool, we conducted statistical studies to define a regression model of ratings against completed activities and to analyse the correlations and covariances between the tool ratings by a user on each scale and the number of activities successfully completed by that user using that tool. Based on the measurement scale results for the different users, we built a parametric regression model for the number of completed activities, using each of the analysed scales as explanatory variables. Table 5 reports the model fitting results.

Adjusted coefficients of the regression model	Human Factors	HCI Factors	Specialization / Functionality Factors	Proposed Scale (Combined Factors)
R	0.979	0.675	0.913	0.989
R2	0.959	0.456	0.833	0.978
R2 adjusted	0.938	0.184	0.750	0.966
SCR	1.233	16.318	5.000	0.672

Table 5. Model fitting results.

As shown in the Table 5, the values of R2 and adjusted R2, indicating the goodness of fit, are closer to 1 (best possible fit) when using the proposed measurement scale combining all the analysed factors, followed by the human factors scale, the specialization / functionality factors and, finally, the HCI factors scale, which has been one of the more popular proposals in recent years and which ultimately proved to be the worst fit for the observed final results of each tool. Therefore, the best scale for predicting the success of any of the tools examined in this paper is the proposed combined factors scale, described in Section 3. The use of the proposed scale to evaluate any tool provides more consistent and reliable results than the other tools analysed in our research.

In view of the high statistical correlation between the completed activities and the measurements (normalized from 0 to 10) of the EUP tool on our joint measurement scale, we conducted an ANCOVA to build a regression model to statistically explain the dependent variable (completed activities) based on the measurements taken on the end users (Table 6).

Evaluation of the value of the information sourced from the variables (H0 = Y=Moy(Y))

Source	DF	Sum of squares	Mean square	Fisher's F	Pr > F
Model	1	20.640	20.659	59528.954	< 0.0001
Residues	99	0.001	0.001		
Total	100	20.763			

Table 6. ANCOVA regression model, fitting results.

The high Fisher's F value, together with Pr>F, both of which are very close to 0, indicate that the explanatory variables used (score on the proposed scale) quite closely explain the success of the above tool for the proposed problem. The model predicts a linear relation between X and Y, and plots the confidence intervals within which 95% of the sample is located as a function of the Y value predicted by the model and the real mean of the observed Y value. The more compact the plot is, the better the model will be, whereas a wider spread denotes that the Y value is more random with respect to X and is not as predictable based on the value of the respective explanatory variable. In sum, the model that provides most information for predicting whether or not an EUD tool will be successful among EUPs is based on the scale proposed in this paper. This model is governed by the following linear adjustment inequality (see Equation 1), from which the success of the tool can be predicted with a probability of 97% (a measurement taken from the adjusted R^2 calculated using the resulting regression model in the statistical study), with a confidence interval of 0.005 in view of the sample, for values of X between the specified values, where X is the final 0-10 rating on our scale:

$$Y \geq +3.5+9.789 * X, \text{ for } X \in [1.8, 8.3] \qquad (1)$$

As we can see, the function relates the group of ratings made by the recruited sample to the group of successful results for the same sample. This piecewise function is not undefined for very low and very high values of X, which predicts a confidence interval for Y, where Y is the percentage of successfully completed activities in the experiment. This fit is only valid where X is greater than or equal to 1.8 and less than or equal to 8.3, as none of measurements made in this experiment had values outside the above range, and it can only be fitted for the specified confidence interval. The value of this function is not intrinsically generalizable, but it is valid for this type of sample, with similar characteristics and background, and the same pre-experimental training (a total of 10 hours of tutorials and workshops) and it does follow a trend in the 100 tests performed (see Figure 4).

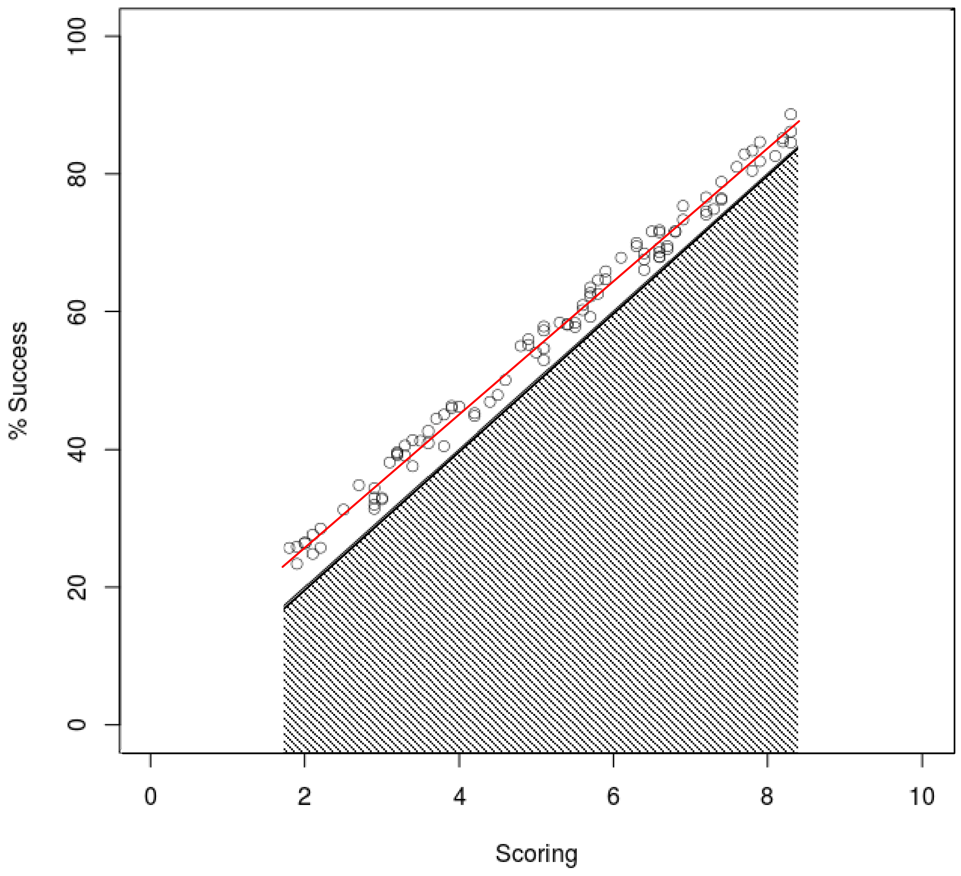


Figure 4. Plot summarizing the x - “rating” and y – “percentage of correctly completed activities” variables for all 100 users and four tools

There is a proven linear relationship (the straight line plotted in Figure 4, which illustrates Equation 1) between the rating of a tool on our proposed scale and the percentage of successfully completed activities by the tool end users. This at least guarantees that tool improvement taking into account the measured factors in order to achieve a better rating on the proposed scale will result in a better performance of the EUP activities using the respective tool. In none of the experiments run so far has the success rate Y been less than 10*X at most, where X is the EUP tool rating on our scale, as illustrated by the shading in Figure 4, corresponding to the function Y = 10*X, the lower bound of the respective confidence interval for the model.

Finally, having analysed RQ1 and RQ2, we conducted a study (see [43]) on the correlation between the tool use success rate and the characteristics of the users that participated in the experiment. This study was again carried out using an ANCOVA (where the percentage of activities was the explained variable and the subject's sex, age, training, employment and experience were the explanatory variables. The results did not provide any statistically significant evidence with respect to any particular factor. The factor with the biggest impact on the explained variable appears to be subject's previous experience in development technologies such as web mashups or PHP (with a R^2 equal to 0.567), followed by age (R^2=0.421), which are nowhere near significant values close to 1 for $\alpha = 0.01$. The conducted study is, at any rate, sound, since the subsample in each group was unbiased with respect to these characteristics, as explained in Section 7.

5. Defining an architecture that adopts the success factors analysed on the proposed scale

In this section, we propose a new architecture for end user-centred service front-end development tools that is based on the proposed measurement scale and addresses each and every one of the success factors covered by this scale. This architecture was devised in accordance with the presented guiding principles, and applies the human, HCI and specialization / functionality factors of the proposed scale. This whole complex architecture has been researched, developed and evaluated as part of several major R&D Spanish and EU-funded projects like EzWeb, FAST, and, more recently, FI-WARE and FI-CORE (two of the biggest European R&D projects funded by the European Commission under its 7th Framework Programme as part of the Future Internet Public-Private Partnership -FI-PPP- initiative), with the aim of building this architecture into EUD tools so that EUPs can more effectively create their own software solutions to meet their requirements [21]. As a result of this effort, we have developed the FAST-Wirecloud mashup tool [27][39]. FAST-Wirecloud has been successfully used in some recent R&D projects, as reported, for example, in [40, 41]. Moreover, FAST-Wirecloud is the mashup solution provided by the FIWARE platform [36]. The FAST-Wirecloud tool, based on architectural components detailed here, is described in Section 6.

The description of the proposed architecture for end user-centred service front-end development tools separates the development, semantics enrichment and authoring phases of the service front-end development lifecycle (see Figure 5) to better depict the different roles, components and relationships that make up this architecture.

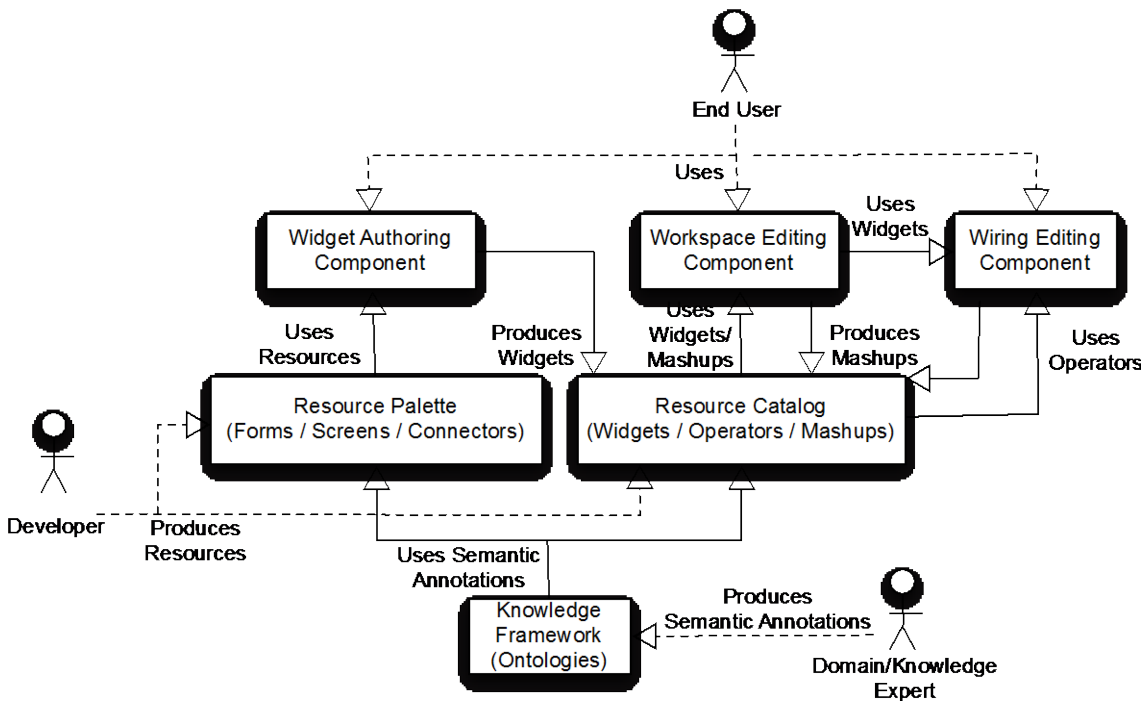

End User
Uses
Widget Authoring Component
Workspace Editing Component
Uses Widgets
Wiring Editing Component
Uses Resources
Produces Widgets
Uses Widgets/ Mashups
Produces Mashups
Uses Operators
Resource Palette (Forms / Screens / Connectors)
Resource Catalog (Widgets / Operators / Mashups)
Developer
Produces Resources
Uses Semantic Annotations
Knowledge Framework (Ontologies)
Produces Semantic Annotations
Domain/Knowledge Expert

Figure 5. Roles, components and relationships in the proposed architecture.

Widgets and operators are the main building blocks of this architecture. A widget represents part of the user interface and application logic necessary to interact with one or more underlying services. Widgets are self-contained front-end components focusing on a single goal and, consequently, are of limited complexity [22]. Widgets can be grouped into workspaces and connected with each other and operators to create a mashup. Operators are similar to widgets, but do not provide a user interface. They are, therefore, intended to implement the necessary application logic to interact with one or more underlying services, and can be used to build less specific widgets. This helps to improve the above specialization/functionality trade-off success factor.

As shown in Figure 5, our proposal is based on providing supporting technology for widgets and operators and for their composition to enable a mashup. The widget will be built by an EUP using the Widget Authoring Component or directly by a software developer. To do this, the EUP may require forms, screens and connectors to back-end resources, which will have to have previously been developed by web programmers and made available in a global Resource Palette. Widgets that have been built and have been tested for correctness by the software provider or, respectively, by the EUP using a verification and validation wizard are published in a global Resource Catalogue, where they can be semantically annotated by a domain/knowledge expert through a knowledge framework. This Resource Catalogue gives end users acting as workspace editors access to these widgets which they use to create the mashup.

In order to better detail our proposed architecture (Figure 5), we explain its three main components:

- A Widget Authoring Component, which is a user-centric IDE dedicated to widget design and creation. Developers can program widgets. Nevertheless, this is a visual tool that helps non-IT-aware users to visually create their own widgets [23] from a palette of composable authoring resources, including widget screens, flows and connectors to off-the-shelf back-end resources (e.g. web/REST APIs), which have been developed by programmers. Two key architectural concepts of this tool are the input and output endpoints associated with widgets, by means of which they can be composed as described below. The tool also offers support for the domain/knowledge experts in order to semantically annotate these artefacts. This component is useful for satisfying the specialization/functionality requirements for users to build the components that they require.

- A Workspace Editing Component intended to design customized user workspaces, like a mashup editor. This tool enables the visual design, reuse and sharing of user workspaces by selecting, connecting and composing the most suitable widgets for dealing with a domain problem in a dashboard [24]. The ultimate aim is to allow end users to create new service front-ends for their specific (situational) needs (instant applications) by visually combining smaller parts (widgets and operators). Each user can have and share any number of workspaces with other members of the community. This component is useful for satisfying the success factors related to interaction between users and the compositional system.

- A Wiring Editing Component that provides a mechanism to visually compose a fully-fledged mashup from the widgets placed in a workspace using the Workspace Editing Tool, which can now interact with each other via events and data sharing. This mechanism is what the composition model calls wiring. The idea behind wiring is easy: widgets display (data/event) inputs and (data/event) outputs, so that, if they are semantically compatible, an output from one widget can be linked to other widgets' inputs. This way, the mashup tool manages the data/event flow between widgets. The mechanism enables the use of event-driven programming features, e.g. a widget can send an event through one of its outputs on an event trigger. Alternatively, the wiring can be automated by following a composition technique based on a pre- and post-condition mechanism described in [47].

Even though the three components (Widget Authoring, Workspace Editing, Wiring Editing) target EUPs, the Widget Authoring Component targets more specialized, possibly corporate EUPs aimed at populating the Resource Catalogue, whereas the Workspace Editing Component and the Wiring Editing Component target EUPs interested in using widgets and operators from the Resource Catalogue as building blocks to build a mashup. Widget authoring cannot be considered an essential, or necessary, part of the process of creating a mashup, and widgets are usually provided directly by application providers. Nevertheless, it is considered here for the sake of completeness, since it targets EUPs.

A mashup is built at two different abstraction levels:

- At *Screen Level*, the EUP visually creates a workspace (e.g. a visualization dashboard or an operations cockpit) by picking out, positioning and resizing the right widgets from a shared catalogue of web components in order to create the desired layout. To do this, he or she uses the Workspace Edition Tool.

- At *Wiring Level*, the EUP visually specifies the relationships between the widgets used at Screen Level and between these widgets and the respective operators (which are also selected from the above-mentioned catalogue) to achieve the desired mashup behaviour. To do so, the EUP visually connects (wires) them. Each connection represents a data/event flow. In the wiring process, the EUP validates the result against to the semantic annotations carried out previously by a domain/knowledge expert on the resources palette and the resources catalogue. Alternatively, the wiring can be automated by means of the pre- and post-condition/fact method explained later in this section.

Having created the workspace and wired the respective widgets and necessary operators, it is published as a new mashup in the Resource Catalogue, which is useful as an asynchronous tool for collaboration between the different roles involved in the development of the end-user solution.

Finally, at run time, an end user can pick and run one of the above workflows on a mashup tool like FAST-Wirecloud, which he or she will have previously adapted to his or her needs with the help of the Workspace Editing Component and the Wiring Editing Component. The proposed architecture enables interoperability between tools from different providers used at mashup design and run time. This interoperability is based on the standardization of the resources published in the catalogue and the tool APIs. Particularly, the specification of the architecture described here, including the catalogue resources (i.e. widgets, operators and mashups) and the Workspace Editing Component and the Wiring Editing Component APIs are part of the FIWARE platform open specifications (see [38]). This, in turn, fosters the applicability of the architecture models for the design of new tools by third parties: the availability of such open specifications "operationalize" these models, which can now be fruitfully exploited by other teams/researchers to design tools that comply with the identified EUD success factors by adopting the proposed architecture.

Figures 6 and 7 describe the FAST and Wirecloud structures used to create the widget-based composite applications that are the result of applying the architecture described in Figure 5 and have been designed considering the success factors. Figure 6 (FAST) focuses on the web components that enable an EUP to design a widget based on its building blocks and a set of facts, whereas Figure 7 (Wirecloud) focuses on the components that are capable of designing a workspace (i.e. a mashup) based on widgets and operators.

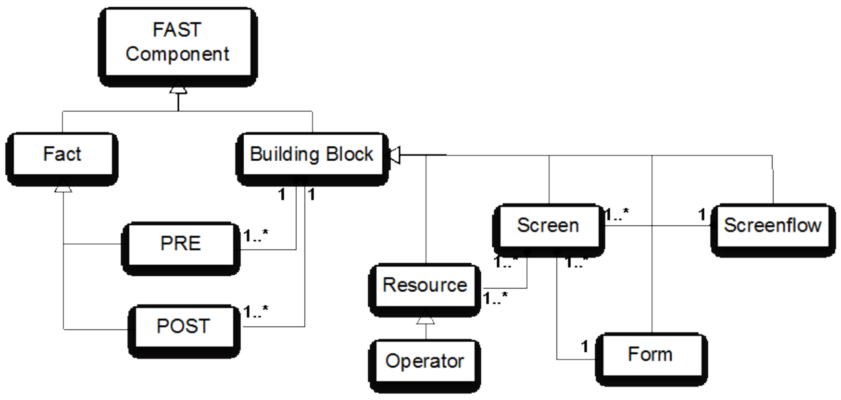


Figure 6. Pre- and post-condition / FAST framework.

As illustrated in Figure 6, there are two types of components at the disposal of a designer to create a widget: facts or building blocks. There are four types of building blocks listed in decreasing order of generality: resources, forms, screens, operators and screenflows. A screenflow is a flow of screens that is valid according to the screen facts. The composition of one or more screenflows results in a widget. A screen is a form that has been connected to an operator in order to provide access to one or more back-end resources. The forms, on the other hand, are generic screens (in the sense of HTML forms) that have not yet been connected to specific back-end resources. These resources may be software provider services or web APIs that have adopted a SaaS philosophy or simply data or list operators (filters, arithmetic operations, data source concatenations, etc.). This hierarchy meets the needs revealed by our EUD success factor scale, where the compositional model has to be adapted to the user cognitive model, abstraction level and provide different specificity/functionality relations. Accordingly, the resulting building blocks become more specific and less generic as the designer composes resources as forms, forms as screens and screens as screenflows to produce a widget.

The pre- and post-condition and facts mechanism is useful for providing guidance to the EUP on the design of a widget based on its building blocks. A fact is a constraint on the data types that a building block accepts or produces. As the EUP designer is not a programmer, this abstraction manages the data types, whereby the designer can build valid data flows between components based on the tool recommendations. These data type constraints build pre- and post-conditions into the managed building blocks. Accordingly, if the pre-condition for component execution are inputs requiring a specific data type, the resources palette can recommend new components that produce such data. Thus, it can recommend new components that consume data generated by the ongoing design completed so far by the designer.

Figure 7 describes the Wirecloud wiring architecture by means of which the EUP can create a mashup.

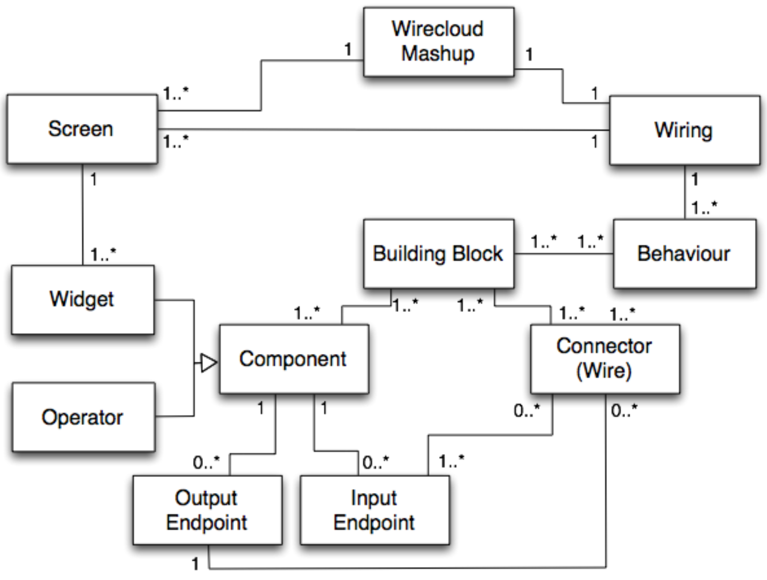


Figure 7. Wiring framework/Wirecloud.

As mentioned above, the wiring model enables the end user to visually connect the widgets taken from the catalogue with each other and to the necessary operators within a screen (i.e. workspace) in order to compose a fully-fledged web mashup in which these components can now interact with each other via events and data sharing. To do this, widgets and operators (the two possible components) display data/event input endpoints and data/event output endpoints, so that an output endpoint from one widget/operator can be linked to other the input endpoints of other widgets/operators by means of wires. The model adopts the concept of behaviour in order to facilitate wiring. Behaviours are useful for partitioning the mashup widget and operator wiring process. Each behaviour accounts for a separate part of the mashup functionality which abstracts a specific mashup “use case” interaction that can be described in natural language by a simple sentence.

Applied to the wiring architecture, that is, assigning pre- and post-conditions and facts to the widget endpoints, the above pre- and post-conditions and facts mechanism is also useful for automating the workspace creation process (i.e. the mashup).

The aim of this architecture (explained in more detail in [20]) is to try to meet the requirements measured objectively on the scale proposed in this paper. With respect to the successful human factors, Table 7 specifies how the architecture includes the three factors that directly influence real end use. The results of the statistical study reported in Section 7, which found that there was an improvement of roughly 20% in each human success factor studied with respect to the tools studied in Sections 2 and 4 confirms this point.

Perceived ease of use	The architecture is defined on the basis of a rather small set of techniques and visual notation elements and quite a simple methodology that minimizes the tool learning curve and helps the user to infer other options once he or she has started to use the techniques and components. Additionally, the EUP can execute the mashup that he or she is building at any time during the process and gather feedback on how he or she is doing. This increases the degree to which an EUP believes that tool use would be free of effort. Besides, the divide and conquer approach to the wiring process introduced by behaviours helps the EUP to build and test the mashup incrementally. This increases the perception of ease of use. The same applies for an EUP who is trying to understand the functionality provided by an existing mashup by directly seeing how each of its parts works and checking when each piece of functionality matches its respective natural language description. Finally, as the number of users increases thanks to the availability of a resources palette and a resources catalogue where both developers and EUPs can share their developments, the tools become more useful.
Perceived usefulness	Mashups typically serve a specific situational (i.e. immediate, short-lived, customized) need. This "situationality" means that they cannot be offered as 'off-the-shelf' functionality by solution providers or IT departments. This creates the need for tools that empower EUPs to create the timely special-purpose software they need to improve their job performance. In doing so, the FAST-Wirecloud tool is automatically perceived by users as useful for improving their job performance.
Computer enjoyment	The ability of and ease with which EUPs using this architecture can pick out the necessary building blocks from an off-the-shelf catalogue of resources to visually compose a operational mashup that will improve their job performance, and the option of being able to run the mashup from the very beginning in order to gather early feedback on how they are doing, is immediately satisfying for EUPs using the tools.

Table 7. How to adopt human success factors

Table 8 summarizes how the proposed architecture accounts for each and every one of the considered HCI success factors.

Abstraction gradient	The notion of behaviour, which EUPs can use to partition the mashup widget wiring (i.e. connection) process into separate parts of mashup functionality, each abstracting a specific “use case” interaction with the mashup that can be described in natural language by a simple sentence, provides the EUP with an adequate level of abstraction when visually creating the mashup. Each snippet is then encapsulated and uniquely identified for later reference and reuse. Furthermore, an EUP can also take advantage of this architectural feature when inspecting other mashups since it provides a modular description of the mashup functionality. Thus the EUP separately inspects the constituent behaviours of the respective mashup to understand its overall functionality (which is a kind of divide and conquer approach to mashup building/analysis). This is also useful for verification and validation purposes. The lowest level of abstraction is the mashup considered as a single behaviour, whereas the top level of abstraction is the partitioning of a mashup into a number of behaviours that can be encapsulated.
Consistency	The architecture is defined on the basis of a rather small set of techniques and visual notation elements and quite a simple methodology that minimizes the tool learning curve and helps the user to infer the other options once he or she has taken the live tutorial provided by each tool. This has been recurrently demonstrated in each and every experiment and study that we have conducted so far.
Error-proneness	The design of the visual notation, which includes support for the semantics-aware approach to the wiring process, helps to avoid careless mistakes. When an EUP wants to connect (i.e. wire) an output endpoint of a given widget to an input endpoint of another widget, detailed information is displayed for the EUP about which endpoints are compatible, incompatible, or even partially compatible with the respective endpoint, requiring, in the latter case, the connection to be made at a given sub-endpoint level . The workspace editing tool itself provides a safeguard against error-proneness since each and every widget in a workspace is isolated from the others by design. All their interdependencies are generated through the wiring process.
Hidden dependencies	Neither the widget authoring component, nor the workspace editing component has hidden dependencies. There could be hidden dependencies between the different behaviours that make up the wiring in the wiring editing component if they share relationships. The editor automatically manages such dependencies by keeping track of the number of occurrences of a given relationship throughout all the behaviours within the wiring. This is maintained until the EUP removes the last occurrence. Any given relationship between two widgets (or between a widget and an operator) can be safely removed without affecting their proper operation.
Premature commitment	Since the workspace editing tool and the wiring editing tool can initially develop (and use) a mashup for a small number of widgets and behaviours, respectively, which are then gradually enhanced by adding new widgets

	and/or behaviours one at a time to adopt additional functionalities, there is no need for the EUP to make decisions before he or she has the all the right information in place. Besides, the widgets and behaviours that are already on board can be easily modified at any time as new requirements are put in place.
Progressive evaluation	A mashup can be executed at any time during its creation process, showing the partial functionality already achieved through the widgets in the dashboard and their wiring. This way, the EUP can find out how he or she is doing at any time. Besides, the behaviour-oriented approach to the wiring enables the EUP to execute the mashup and check whether he or she gets the specific functionality described by each and every one of the behaviours of which it is composed. This eases the task of determining when the overall mashup functionality is achieved. It also helps an EUP to better understand the functionality provided by a given behaviour by directly seeing how it works and checking when this functionality matches its respective natural language description.
Role expressiveness	The EUP can see how each mashup component relates to the whole by means of the wiring editor, which shows the existing relationships (data communication and events) between each component (widgets, operators) and the rest of the mashup. Besides, the EUP can even see how each specific interaction use case (i.e. each behaviour) relates to the whole functionality of the mashup.
Viscosity	It is relatively effortless to make a single change. All you have to do is either add/remove/resize/reposition the necessary widgets in the workspace, or add/remove the necessary components (widget, operator and/or wire) to the respective behaviour, or create a new behaviour if needed.
Visibility and juxtaposability	Each and every part of the development (i.e. the mashup) is simultaneously visible in the wiring editor, and the EUP is capable of focusing on meaningful parts of the mashup through the behaviour editor so that he or she can compare any two parts and their relationship side by side.

Table 8. How to adopt HCI success factors

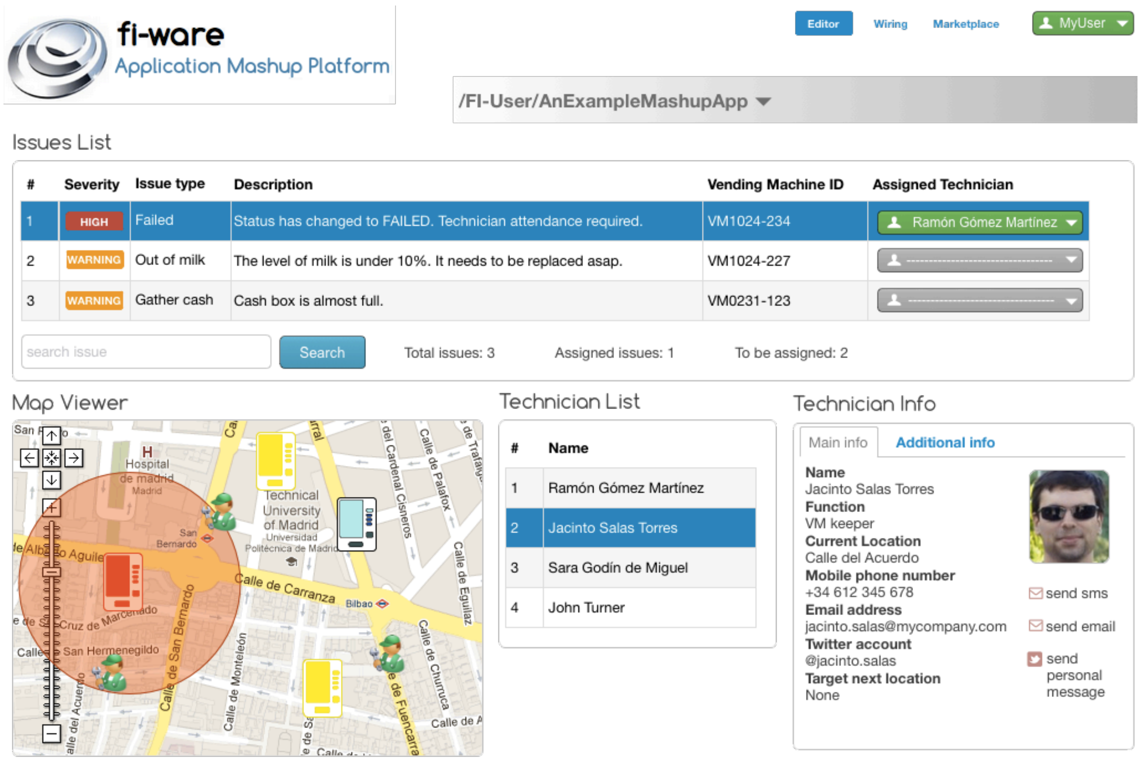
With regard to the specialization and functionality factors, our architecture offers a rather good trade-off between the level of specialization and the functionality of the created solutions. This is achieved thanks to the fine-grained modularity offered by the components available in the catalogue (widgets and operators) and the level of configurability that they offer for customization. Thus they can be used in mashups that are very specialized for a given problem, but can at the same time be easily exported and used in other application domains. As a result, the EUP can start from a given mashup specialized for a given domain and change it by reconfiguring its components (through parameterization) and/or by reconnecting them to other (new) components. The shared repository (catalogue) of components (widgets, operators and other existing mashups) actually offers separate sets of components for different application domains, along with a number of general-purpose, cross-domain components. In particular, the use of operators, which are intended to implement the application logic necessary to interact with one or more underlying services, provides for more generic widgets, which can therefore be more application logic-agnostic (i.e. domain-independent) and focus on providing the user interface logic.

Additionally, the availability of separate sets of components for different application domains in the resource catalogue, and the separation of concerns between widgets and operators (where operators take care of most of the specificities of the application domain) also helps to cater for domain specificity (in the sense of [46]), i.e., how customizable the mashup tool is for specific requirements possibly emerging in specific domains.

6. The FAST-Wirecloud mashup tool implementing the proposed architecture

FAST-Wirecloud is a mashup tool in which widgets published in a collaborative catalogue can be interconnected and arranged in one or more workspaces to create an application mashup that satisfies instant requirements. The FIWARE Academy website offers training courses, lessons and many other contents that demonstrate FAST-Wirecloud features and teach EUPs how to create their own solutions. In doing so, FAST-Wirecloud implements the architecture proposed by the authors in the previous section. In particular, FAST implements the Widget Authoring Component (i.e. the pre- and post-condition / fact framework), whereas Wirecloud implements the Workspace Editing Component and the Wiring Editing Component (i.e. the wiring framework).

Figure 8 illustrates the FAST-Wirecloud Workspace Editing Component, which enables the EUP to build an application mashup from a number of widgets. Once the widgets are placed in the workspace, it can be automatically executed, because the FAST-Wirecloud Workspace Editing Component is also the mashup tool runtime component. This goes along with the findings of some related work in the literature, which has found that the distinction between the mashup tool design or authoring phase and execution phase is not perceived as effective by EUPs (see, for example, [45]). Besides, this gives the user the chance to see how the designed widget works both during its design in the Widget Authoring Component and later when it is integrated into a workspace with the Workspace Editing Component to compose the application mashup. This provides immediate feedback for testing the validity of the composition.



Issues List

#	Severity	Issue type	Description	Vending Machine ID	Assigned Technician
1	HIGH	Failed	Status has changed to FAILED. Technician attendance required.	VM1024-234	Ramón Gómez Martínez
2	WARNING	Out of milk	The level of milk is under 10%. It needs to be replaced asap.	VM1024-227	--------------------
3	WARNING	Gather cash	Cash box is almost full.	VM0231-123	--------------------

search issue | Search | Total issues: 3 | Assigned issues: 1 | To be assigned: 2

Technician List

#	Name
1	Ramón Gómez Martínez
2	Jacinto Salas Torres
3	Sara Godín de Miguel
4	John Turner

Technician Info

Main info | Additional info

Name
Jacinto Salas Torres
Function
VM keeper
Current Location
Calle del Acuerdo
Mobile phone number
+34 612 345 678
Email address
jacinto.salas@mycompany.com
Twitter account
@jacinto.salas
Target next location
None

send sms
send email
send personal message

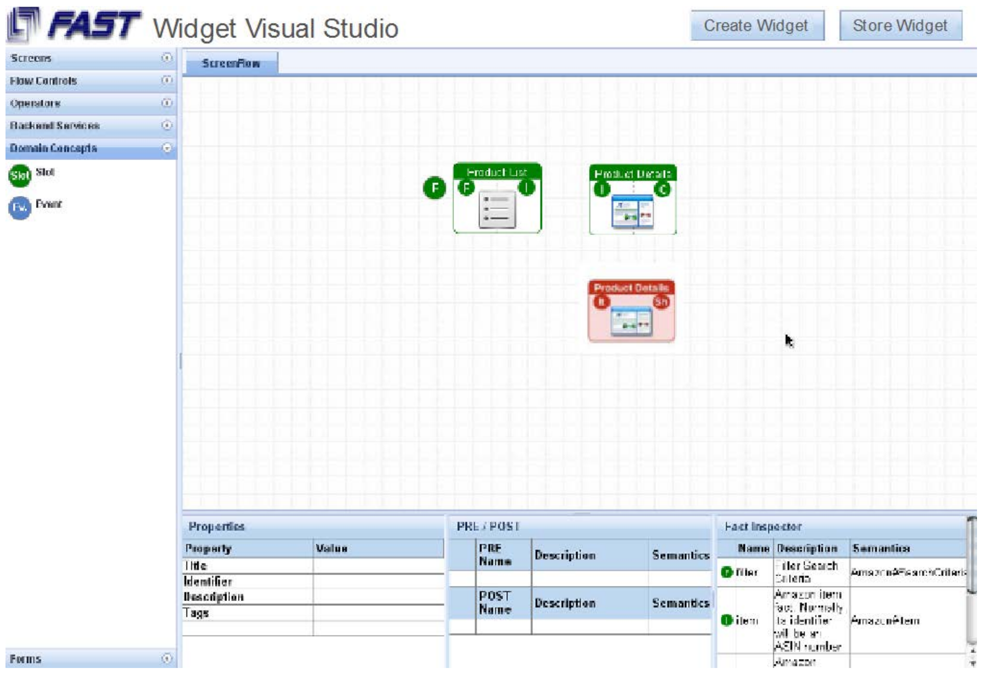
Figure 8. FAST-Wirecloud Workspace Editing Component.

However, if EUPs are unable to find what they need in that catalogue, they can create new widgets using the FAST-Wirecloud Widget Authoring Component, designed to enable non-programmer users to create widgets from more specific components called resources, available in public catalogues and around Internet.

The FAST-Wirecloud Workspace Editing Component incorporates the Consistency, Error proneness, Premature Commitment, Progressive Evaluation, Viscosity and Visibility success factors.

Figure 9 illustrates the FAST-Wirecloud Widget Authoring Component, which strictly implements the pre- and post-condition / fact framework proposed in this paper. Figure 9 shows three screens that are being integrated into a screenflow. The circles denote the pre- and post-conditions and are coloured green or red depending on whether or not the associated conditions are met. Note that the Product Details screen cannot be integrated because its pre- and post-conditions are coloured red. To be able to integrate this screen, the user must add one or more other screens to the screenflow to assure that the pre- and post-conditions are met. The tool will give the user recommendations for this purpose.

By design, the FAST-Wirecloud Widget Authoring Component incorporates the same success factors as the FAST-Wirecloud Workspace Editing Component did, i.e. Consistency, Error Proneness, Premature Commitment, Progressive Evaluation, Viscosity and Visibility. Nevertheless, as stated above, it targets a different, more specialized (and possibly corporate) user aimed at populating the resource catalogue with ready-to-use widgets.



Property	Value
Title	
Identifier	
Description	
Tags	

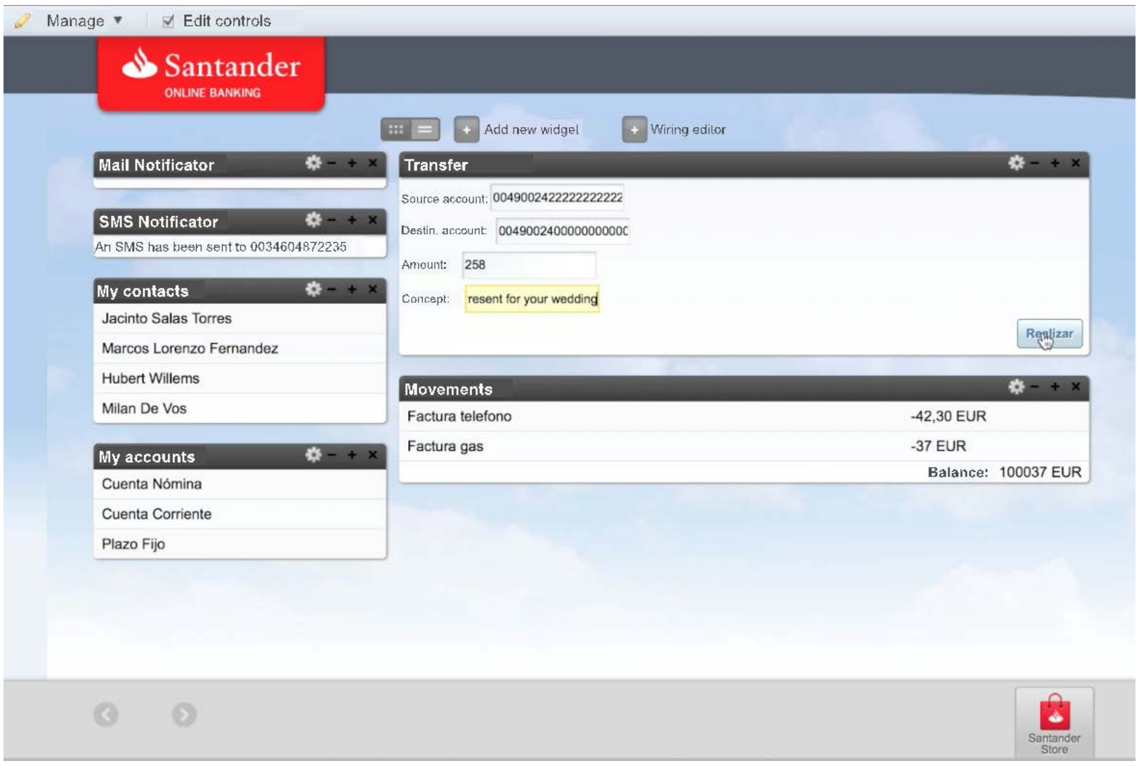
PRE Name	Description	Semantics

POST Name	Description	Semantics

Name	Description	Semantics
filter	Filter Search Criteria	AmazonAPI#searchCriteria
item	Amazon item fact. Normally its identifier will be an ASIN number	Amazon#Item
	Amazon	

Figure 9. FAST-Wirecloud composition component.

To better illustrate the wiring process, we borrow a common scenario from the banking domain: wire transfers over Internet. The Centre for Open Middleware [48], a Santander and UPM Joint Technology Centre, developed the widgets and the mashup used in this scenario as a case study for the open source implementation of the proposed architecture built using Java and Liferay, as a proof of concept. As shown in Figure 10, the resulting mashup contains two widgets that display the list of contacts and the list of accounts, respectively, along with a widget that shows the details of a given account and a widget that enables the user to enter the wire transfer details. Additionally, the mashup includes a couple of additional widgets that enable the user to check the result of the notification of a given transfer via email and SMS.



Mail Notificator

SMS Notificator
An SMS has been sent to 0034604872235

My contacts
Jacinto Salas Torres
Marcos Lorenzo Fernandez
Hubert Willems
Milan De Vos

My accounts
Cuenta Nómina
Cuenta Corriente
Plazo Fijo

Transfer
Source account: 0049002422222222222
Destin. account: 004900240000000000C
Amount: 258
Concept: resent for your wedding

Movements

Factura telefono	-42,30 EUR
Factura gas	-37 EUR
Balance:	100037 EUR

Figure 10. FAST-Wirecloud wiring model.

Obviously, the widgets in the mashup are of no use until they are interconnected. The Movements widget can be connected to the My Accounts widget to show the details of the account selected in the widget. The Transfer Details widget can be connected to My Accounts and My Contacts for auto-completing the source and target account fields. Finally, the SMS and email notifiers can be connected to My Contacts to send the alert to the contact selected in the My Contacts widget.

Figure 11 illustrates the application of the FAST-Wirecloud wiring model to this scenario, which is a strict implementation of the wiring framework of the reference architecture proposed in this paper.

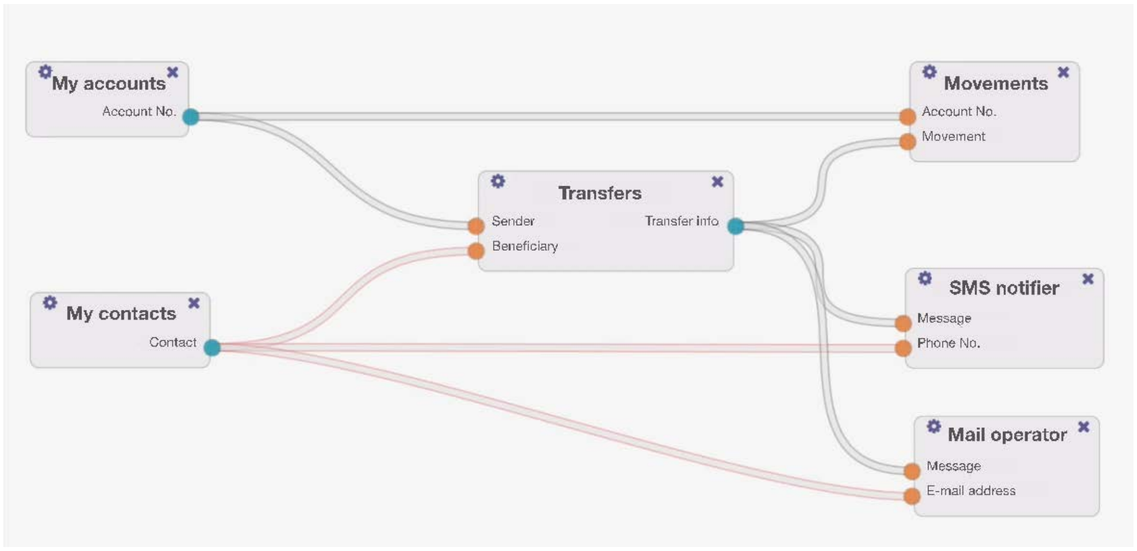


Figure 11. FAST-Wirecloud wiring model applied to the wire transfers scenario.

The wiring editor takes the end user through the process of wiring the widgets that make up the mashup. It visually advises the end user on which endpoints could be wired by highlighting the endpoints that are semantically compatible (i.e. connectable) in green. Figure 12 below illustrates this idea. The Sender and the Beneficiary endpoints of the Transfers widget are semantically compatible with the Account No. endpoint offered by both the My Accounts and Movements widgets.

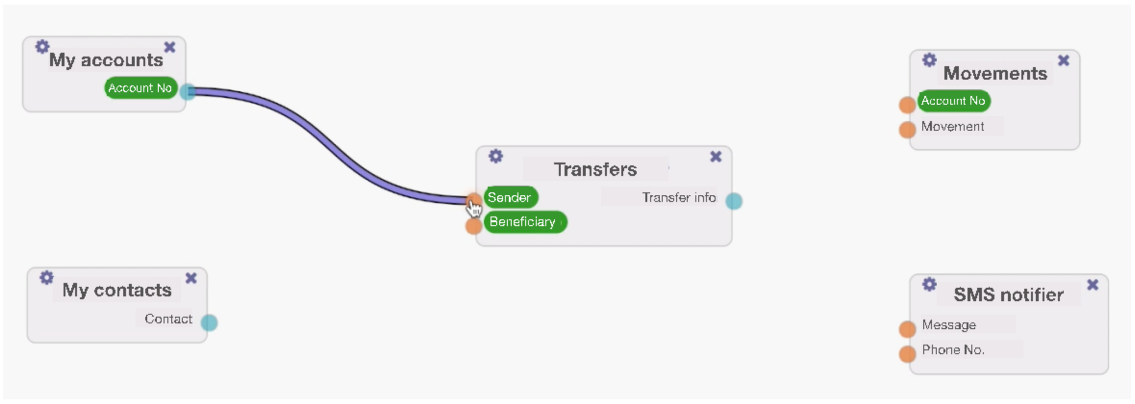


Figure 12. FAST-Wirecloud wiring model: Semantic-awareness feature.

This feature even works at sub-endpoint level, i.e. can recursively analyse the data structure that is accepted/sent by each endpoint. Figure 13 illustrates this idea. On the left side, the EUP is informed that the Contact endpoint of the My Contacts widget could be connected to the Beneficiary endpoint of the Transfers widget, but not directly (the endpoint is highlighted in amber, instead of green). This way, the EUP can inspect the structure of the Contact endpoint and decide which sub-endpoint to use (if there is only one compatible sub-endpoint, it can be wired automatically without having to work at sub-endpoint level).

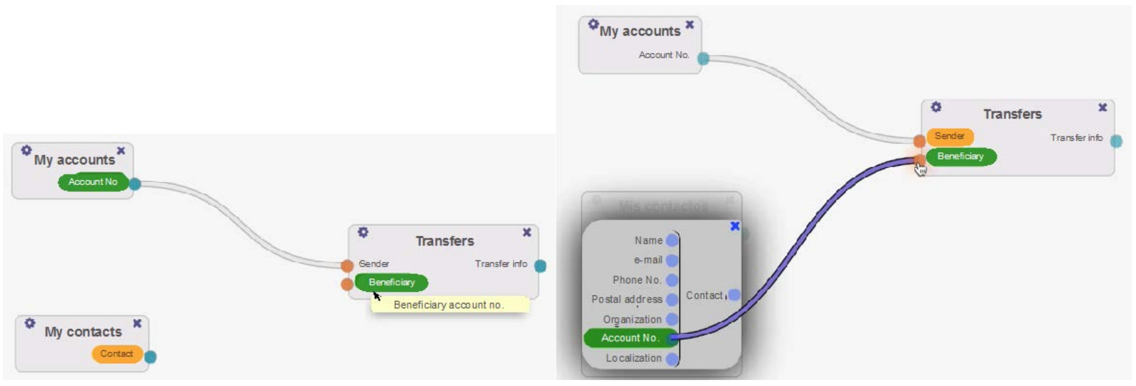


Figure 13. FAST-Wirecloud wiring model: Semantic-awareness feature.

This feature uses an ontology server and requires widgets and operators to be annotated with the elements of at least one domain ontology and more than one domain-agnostic high-level ontology, but the EUP is isolated from these formalisms and is presented with a user-friendly visual metaphor.

The tool also offers a feature that enables the EUP to partition the wiring into different behaviours, following the wiring framework concepts described in Section 5. Figure 14 shows a mashup whose wiring is made up of two different behaviours. The first behaviour (top) describes the wiring needed to make a video call to a given technician chosen by the operator. The second behaviour (bottom) describes the wiring needed to display the profile and the location of all the available technicians at any time.

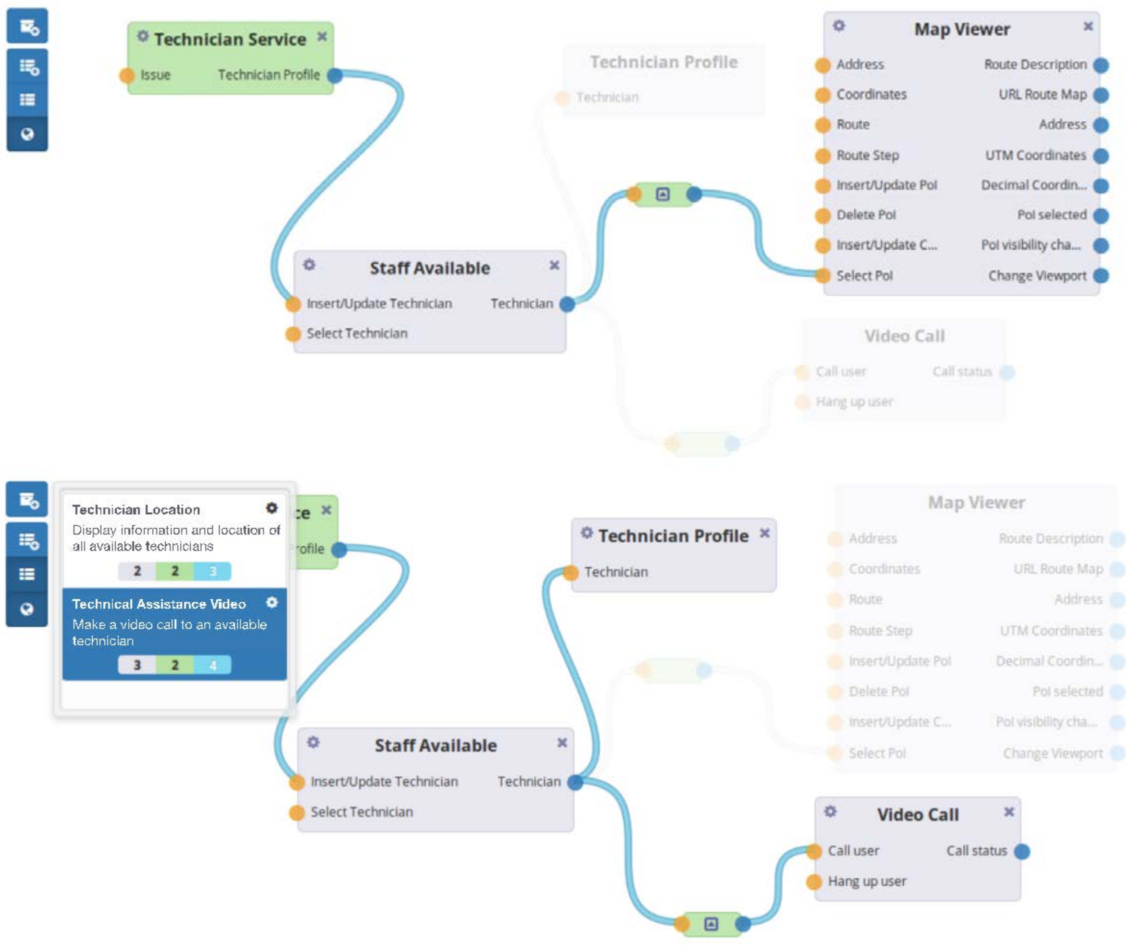


Figure 14. FAST-Wirecloud wiring model: Behaviour-oriented partitioning feature.

As shown in Figure 14, the EUP does not need to deal with the whole wiring graph (describing the overall mashup) at once. The widgets, operators and connections belonging to the behaviour selected by the EUP at any time are highlighted so that he or she can focus on whichever he or she requires. At the same time, however, the EUP always has an overview of the entire mashup so that he or she can pick out any widget/operator/connection already in use (i.e. present in one or more behaviours) for addition to the current behaviour, if necessary.

Any given connection can belong to more than one behaviour. In this case, it is only dropped from the overall mashup when it is dropped from all the behaviours (i.e. from the last behaviour of which it was part).

The FAST-Wirecloud Wiring Model incorporates the Abstraction Gradient, Consistency, Error Proneness, Premature Commitment, Progressive Evaluation, Role Expressiveness, Viscosity and Visibility and Juxtaposability success factors.

In order to evaluate the goodness of fit of the success factors to the proposed architecture and the mashup platform, we conducted another statistical experiment which is described in Section 7 below.

7. Evaluation of the proposed tool using the EUD success factor scale and an end user-based experiment

Now that we have presented the FAST-Wirecloud composition tool, research focuses on evaluating its use and proving that our premise of enabling EUPs to build their own composite applications is feasible and true. This tool will be evaluated with the proposed measurement scale to ensure that its development based on achieving the EUD success factors effectively translates into a good score on the scale. The tool also includes an advanced wizard that helps EUPs throughout the application development life cycle, but, as this wizard is not adaptable to the other tools, it has been disabled to assure a fair comparison. Finally, we have to check if this hypothetical achieved good score is representative of the real success of the EUPs in our study achieving better results than with existing tools. FAST-Wirecloud evaluation aims to test whether the developed user-centred composition system satisfies its usability, functionality and performance requirements.

The proposed evaluation was the same as stated for Section 4, that is, a new sample of 25 users tackled the proposed problem (and its 100 activities) using FAST-Wirecloud to solve the problem, using the same research questions RQ1 and RQ2 as listed in Section 4. Table 9 shows the characteristics of this new sample.

Characterization		WireCloud/FAST group
Gender		
	Male	13
	Female	12
Age		
	< 20 years	5
	20-34 years	6
	35-49 years	5
	50-64 years	5
	> 65 years	4
Educational Attainment		
	Secondary School	6
	Vocational Training	7
	Bachelor's Degree	6
	Master's Degree	6

Employment		
	Student	7
	Researcher	8
	Employee	10
Experience and previous knowledge		
	Mashup Tools	1
	Web Services (SOAP, ESB, BPEL, etc.)	0
	HTML, CSS	0
	Java, J2EE	0
	JavaScript, AJAX	0
	PHP, ASP	0
	OO Programming	0
	C, C++, C#	0
	Scripting, Perl	0
	Haskell, Prolog	0

Table 9. Characterization of the sample of users that participated in the experiment with WireCloud/FAST.

This new group was validated by means of an ANCOVA study which found that this group was not biased with respect to the sample of 100 users employed earlier. Adding this new subsample to the ANCOVA [43] conducted to validate the first 100 years, the results for R^2 and adjusted R^2 were close to 0.02, far removed from the value 1 that would suggest a possible statistically significant bias. We looked for people with the same profile as the four groups formed in the studies explained above.

The results of new experiments carried out by this new set of 25 users are shown in Table 10.

	Human Factors	**HCI Factors**	**Spec/Functionality Factors**	**Proposed Tool**	Percentage of Resolved Activities
WireCloud/Fast	Mean 6.95 Median 7.20	Mean 19.5 Median 19.3	Optimal	Mean 7.50 Median 7.47	75%

Table 10. Results of the evaluation of the proposed architecture.

Looking at Table 10, the first relevant fact is that the new tool users managed to successfully complete 75% of the activities associated with the problem. Accordingly, a sizeable part of the sample was able to successfully perform the requested compositional development, which contrasts with the poorer results of the other tools. The second issue to be taken into consideration is that FAST-Wirecloud scored higher than all the other analysed tools on all measurement scales, especially on the scale proposed in this paper. This is because the FAST-Wirecloud architecture has been modelled taking into account the factors measured by the respective scales.

Finally, we checked whether the new explanatory data (ratings attached to FAST-Wirecloud) and new study data (number of activities successfully completed by new EUPs) led to changes in the correlation and ANCOVA studies reported in Section 4. We found in Table 11, which includes the statistical data for all five subsamples, that the correlations of the characteristics of these users (sex, age, training, employment, previous experience) with the results are almost identical and do not provide any new statistically significant evidence to suggest that any of them is directly correlated to the results. The complete statistical calculations are reported in [43].

Analysis of variance:

Source	df	Sum of squares	Mean squares	F	Pr > F
Model	35	5.932	0.169	1.196	0.264
Error	89	9.072	0.142		
Corrected Total	124	15.004			

Computed against model Y=Mean(Y)

Type I and Type III sum of squares analysis:

Source	DF	Sum of squares	Mean squares	F	Pr > F	R² of the partial model
Gender	1	0.134	0.134	0.943	0.335	0.331
Age	3	0.752	0.251	0.968	0.262	0.432
Educational Attainment	2	0.163	0.081	0.575	0.566	0.214
Employment	6	0.456	0.076	0.536	0.779	0.114
Experience and previous knowledge	22	4.387	0.199	1.407	0.146	0.578

Table 11. ANCOVA results with the statistical effect of each characteristic on the percentage of completed activities dependent variable.

Note, from Table 11, that the selected explanatory variables cannot be considered to be the source of a significant amount of model information (Pr > F = 0.264 >> 0.01). The model is not significant because these data suggest that the percentage of completed activities is independent of the characterization of the sample. Of the studied variables, the variable with the greatest Fisher F-distribution is previous knowledge and expertise (F=1.407). Pr > F is equal to 0.146 (the closest to 0.01) for that variable. The next variable is again age (as found in the study reported in Section 4). Therefore, we can infer that, again, there is no sign of there being any critical factor in the conducted study.

As the population subsample in each of the five subgroups is unbiased with respect to these characteristics, the two studies reported in Section 4 and Section 7 are, at any rate, sound, although we intend to embark upon a new future line of research to specifically explore the dependence of this type of tools targeting end users on specific user characteristics, as explained in Section 9.

Therefore, evaluating a web development tool on the proposed scale provides both a priori and a posteriori knowledge of how many users would be successful using this tool. According to the nonlinear model calculated above, the number of solutions produced in this case is very similar to expectations considering the resulting score calculated according to Equation (1).

Now that the preliminary evaluation of this tool based on the proposed scale is complete, WireCloud/FAST is undergoing a further evaluation process through its use in FP7 projects. There is qualitative and quantitative evidence that FAST-Wirecloud has been successfully used in some recent R&D projects [40, 41]. Some evidence of the extensive use of this platform follows:

- Downloads of the last version from PyPI: more than 2600 downloads per month [1]
- Use of the FAST-Wirecloud portal instance at FIWARE Lab [42] (http://mashup.lab.fiware.org):
    - Users: 5936 (as of Dec. 2015)
    - Total dashboards developed: 9540 (143 public) (as of Dec. 2015)
    - Total Mashable Application Components (MACs) developed: 2785 (as of Dec. 2015)

A recent survey conducted by the FIWARE Accelerator Programme in mid-2015[36] asked about the perceived usefulness and maturity level of the mashup solution offered by FAST-Wirecloud platform as a tool implementing the details of the success factors explained here. To do this, a five-point Likert scale was used: 1 – Completely immature/useless, 2 –Low maturity, 3 – Mid-maturity, 4 – Mature, final adjustments needed, 5 – Ready for market. The 68 companies that took the survey stated that they were using (33) or planned to use (35) FAST-Wirecloud as an application mashup solution for their products (see http://catalogue.fiware.org/ for an exhaustive lists). With regard to maturity, the surveyed 68 companies that stated that they were using FAST-Wirecloud gave this applications mashup solution a score of "4 – Mature, final adjustments needed" (mean 3.48, mode 4, median 4). The conclusion of the survey is that the international community of users perceives FAST-Wirecloud to be a useful and mature mashup solution.

8. Discussion of threats to validity

This discussion on threats to the validity of end-user development success factors for mashup development environments in EUD will refer to five aspects of validity, which can be summarized as follows:

- Construct validity: This aspect of validity reflects the extent to which the operational measures that are studied really represent what the researcher has in mind and what is investigated according to the research questions. The stated research questions aimed to demonstrate that the results of designing a EUD web tool using particular quality factors are more satisfactory than using current tools, which we believe has been proven in this research. The studies used to respond to the research questions have been carried out with non-biased samples, designed to reflect the target profile of the archetypal EUD tool user. Additionally, ANCOVA and standard regression models were employed to assure statistical data validity. We have taken into account some user characteristics that may later have an impact of the soundness of the measurements, such as sex, age, training, employment and previous experience in software tools (a characteristic that should be and seldom is taken into account in this type of studies). In this study we have found that none of the characteristics influence the analysed RQs, but it is, in any case, essential to assure that there is no bias in the distribution of the subsamples. On this ground, we originally used four groups, later adding a fifth group which is absolutely equivalent with respect to the above characteristics.

[1] https://pypi.python.org/pypi/wirecloud/0.8.4

- Internal validity: This aspect of validity is of concern when examining causal relations. When the researcher is investigating whether one factor affects an investigated factor there is a risk that the investigated factor is also affected by a third factor. If the researcher is not aware of the third factor and/or does not know to what extent it affects the investigated factor, there is a threat to internal validity. In principle, the covariance studies suggest that the studies are not influenced by external factors that have not been taken into account, as all the possible factors and characteristics affecting each user have been accounted for. Additionally, we believe that the reward offered to users for participating in the study (free user accounts for the beta version of FAST and beta licences for all the software presented at, as well as free registration for, the congresses) was proportionate, thereby removing the threat of compensation causing selection bias and potentially invalidating the study.

- External validity: This aspect of validity is concerned with the extent to which it is possible to generalize the findings and how much interest the findings are to other people outside the investigated case. The sample is large enough to suggest that the data are generalizable. The analysed tools are free, and a similar sample can be recruited to replicate the results step by step. Additionally, FAST-Wirecloud has become a Fi-WARE FP7 project reference tool, and has thus been elevated to the standard EUD platform in Europe. On this ground, we believe that its principles and success factors can be extrapolated to other fields.

- Reliability: This aspect is concerned with the extent to which the data and the analysis are dependent on the specific researchers. As shown in the paper, we have conducted statistical studies based on direct observations, questionnaires and specific practical exercises which are not at all subjective.

9. Conclusions

More and more software vendors are embracing the SaaS (software as a service) philosophy and providing all their products, data and services as web services, which are accessible not only to businesses but also to all Internet users [30]. Thanks to this philosophy and the emergence of open data published by government agencies, many consumers and small businesses are able to create more and more complex applications organizing calls between services and calling and remixing various data sources. However, EUPs do not have access to these benefits and privileges.

However, as the Web 2.0 philosophy showed a decade ago, EUPs have to be taken into account as "prosumers", providers of applications and data that they previously could only consume. To achieve this milestone, it is not enough to provide users with user-centric development tools, such as mashup tools. There is statistical evidence that current EUD web tools have little success among users without programming skills. Therefore, it is necessary to study the specific factors that have led other applications (e.g. spreadsheets) to ensure that millions of EUPs become programmers of applications that will solve everyday problems [31]. This article presents research on these factors, and existing measurement scales for their promotion. It also provides a new measurement scale based on existing research, which has proved to be better aligned with what happens when an EUP needs to successfully use a particular tool. As a result of this research, we report a reference architecture, which is based on the success of EUD factors and meets the needs of the EUPs who have tested and used its open source reference implementation, FAST-Wirecloud, as a guideline for improving the existing technology in this field.

FAST-Wirecloud incorporates the three human factors that directly influence the actual end use of a software solution: perceived usefulness, perceived ease of use and computer enjoyment. As it targets specific situational needs that cannot be catered for by traditional "off-the-shelf" applications or by IT departments following a planned development process, FAST-Wirecloud is automatically perceived by its users as useful for increasing their job performance, as shown by the ratings of the items related to human success factors and the comments to open questions as part of the survey conducted at the end of the study. The open responses stated that the option of simply picking the necessary building blocks to visually compose the functional mashup to improve their job performance from an off-the-shelf catalogue of resources, and the option of executing the mashup from the very beginning of its development to gather early feedback on how they were doing were an asset. They also generated immediate satisfaction and the desired ease of use perception among end users. Perceived ease of use is also promoted through the divide-and-conquer approach to the wiring process introduced by the behaviour concept.

With regard to the HCI success factors, the paper has illustrated how the proposed architecture and its reference implementation FAST-Wirecloud promotes the abstraction gradient, consistency, error-proneness, premature commitment, progressive evaluation, role expressiveness, viscosity, and visibility and juxtaposability factors.

Finally, we argue that the proposed architecture and its reference implementation FAST-Wirecloud offer a rather good trade-off between the level of specialization and the functionality of the resulting solutions thanks to the fine-grained modularity offered by the components available in the shared resource catalogue and the level of configurability for customization. This allows for their use in mashups that are very specialized for a given problem, while they can at the same time be easily exported and used in other application domains. In particular, more generic widgets can be built using operators, which can therefore be more application logic-agnostic (i.e. domain-independent). Additionally, the availability of separate sets of components for different application domains in the resource catalogue, and the separation of concerns between widgets and operators (which take care of most application domain specificities) also helps to cater for domain specificity, i.e. customize the mashup tool for specific requirements that possibly emerging in specific domains.

Regarding the future trends of this work, the key line of future research to be undertaken next is to study the web components within the tool catalogues and analyse which metrics they should meet to qualify as quality components, how quality can be assured and how to improve components so that they can be used by EUPs with every guarantee of success. This work will be the next logical step in this research field, considering that we have measured the different success factors to be met by an EUP tool.

Finally, this study has not turned up any correlation between end users' previous experience in the use of software tools and the results for EUD tasks, a correlation that we suspected would exist. Most of the sample (94.4%) had no experience, and the few experienced users that took part were equally divided across the five work groups. On this ground, we were unable to study the impact of this characteristic in more depth. Therefore, another RQ that might be addressed in the future is to analyse the real impact of this type of previous experience on EUD user success. This study could be performed as part of the new qualitative and quantitative analyses on the FAST-Wirecloud tool and the success factors that have driven its development based on the data from hundreds of users that are now using this tool today.

Acknowledgements

This research was partially supported by the European Union co-funded IST projects FAST: Fast and Advanced Storyboard Tools (GA 216048), FI-WARE: Future Internet Core Platform (GA

285248) and FI-CORE: Future Internet - Core (GA 632893). The FI-WARE and FI-CORE projects are part of the European Commission's Future Internet Public-Private Partnership (FI-PPP) initiative.

We would like to thank Fernando Alonso Amo for his support for and collaboration in the reported research, which has been very helpful for drafting this paper.

References

[1] G. Alonso, F. Casati, H. Cuno, and V. Machiraju, Web Services Concepts, Architectures and Applications. Data-Centric Systems and Applications. Germany: Springer, 2004.

[2] C. Schroth and O. Christ, "Brave new web: Emerging design principles and technologies as enablers of a global SOA," in Proceedings of the IEEE International Conference on Services Computing, 2007. SCC 2007. Los Alamitos, CA (USA), 2007, pp. 597–604.

[3] D. Lizcano, J. Soriano, M. Reyes, and J. J. Hierro, "Ezweb/FAST: Reporting on a successful mashup-based solution for developing and deploying composite applications in the upcoming "ubiquitous SOA"," International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, September 29 - October 4, Valencia, Spain. Pp. 488–495, 2008.

[4] P. J. Molina, I. Torres, O. Pastor, "User Interface Patterns for Object-Oriented Navigation", in Upgrade, Human-Computer Interaction: Overcoming Barriers Vol. 4, issue 1, February 2003.

[5] A. P. McAfee, "Enterprise 2.0: The dawn of emergent collaboration," MIT Sloan Management Review, vol. 47, no. 3, pp. 21–28, 2006.

[6] A. S. Lee, "Remarks from MIS quarterly editor – inaugural editor's comments," MIS Quarterly, vol. 23, no. 1, 1999.

[7] R. Smith, "Enterprise mashups: an industry case study," in Keynote at New York PHP Conference and Expo. NY, USA: IBM Software Group Press, June 2006.

[8] C. Anderson, The Long Tail: Why the Future of Business Is Selling Less of More. NY, USA: Hyperion, July 2006.

[9] T. Janner, R.Siebeck, C. Schroth and V. Hoyer, "Patterns for Enterprise Mashups in B2B Collaborations to Foster Lightweight Composition and End User Development", in Proceedings of the 2009 IEEE International Conference on Web Services (ICWS '09), Los Angeles, CA, USA, pp. 976-983, 2009

[10] J.H. Wu, Y.C. Chen, and L.M. Lin, "Empirical evaluation of the revised end user computing acceptance model," Computers in Human Behavior, vol. 23, no. 1, pp. 162 – 174, 2007.

[11] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy and R. and Selby, "Cost models for future software life cycle processes: COCOMO 2.0," Annals of software engineering, vol. 1, no. 1, pp. 57-94, 1995.

[12] C. Scaffidi, M. Shaw and B. Myers, "Estimating the numbers of end users and end user programmers," in IEEE Symposium on Visual Languages and Human-Centric Computing, Pittsburgh, USA, September 20-24, 2005.

[13] A. Blackwell and T. R. G. Green, “Investment of Attention as an Analytic Approach to Cognitive Dimensions, in Collected Papers of the 11th Annu. Workshop Psychology of Programming Interest Group (PPIG-11), T. R. G. Green, R. H. Abdullah & P. Brna, Eds. , Leeds, UK. pp. 24-35, 1999,

[14] S. P. Jones, A. Blackwell, and M. Burnett, “A user-centred approach to functions in Excel,” in ICFP 03: Proceedings of the eighth ACM SIGPLAN international conference on Functional programming. Uppsala, Sweden, 2003, pp. 165–176.

[15] H. Lieberman, F. Paternò, M. Klann, and V. Wulf, End-User Development: An Emerging Paradigm. Human-Computer Interaction Series. Germany: Springer, Nov. 2006, vol. 9, pp. 1–8.

[16] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, “Extrinsic and intrinsic motivation to use computers in the workplaces” Journal of Applied Social Psychology, vol. 22, no. 14, pp. 1111–1132, 1992.

[17] M. Gaedke, “Web Engineering: Creating Solutions in the Age of Emotion, SOA, and Web 2.0”, Tutorial in the 17th International World Wide Web Conference, Beijing, China, April 20, 2008.

[18] T. Green and M. Petre, “Usability analysis of visual programming environments: A cognitive dimensions framework,” Journal of Visual Languages and Computing, vol. 7, no. 2, pp. 131–174, 1996.

[19] S. Meliá, J. Gómez, S. Pérez, O. Díaz, “Architectural and Technological Variability in Rich Internet Applications”, Internet Computing, IEEE, vol.14, no.3, pp.24-32, 2010.

[20] D. Lizcano, F. Alonso, J. Soriano, G. López, End-User Development Success Factors and their Application to Composite Web Development Environments, The Sixth International Conference on Systems (ICONS 2011), St. Maarten, The Netherlands Antilles, January 23-28, 2011.

[21] V. Hoyer, A. Fuchsloch, S. Kramer, K. Moller, and J. López, “Evaluation of the implementation,” FAST Consortium, Tech. Rep. D6.4.1, February 2010.

[22] B. Shneiderman “Promoting universal usability with multi-layer interface design”, in Proceedings of the 2003 conference on Universal usability, 1-8, New York, NY, USA.

[23] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” Ph.D. dissertation, University of California, Irvine, 2000.

[24] J. Wong and J. I. Hong, “Making mashups with marmite: towards end-user programming for the web”, in Proceedings of the SIGCHI conference on Human factors in computing systems; 1435-1444, New York, USA, 2007.

[25] Z. Obrenovic and D. Gasevic, “Mashing up oil and water: Combining heterogeneous services for diverse users”, in IEEE Internet Computing, 13, 6, pp. 56-64, 2009.

[26] EzWeb (2012) http://demo.ezweb.morfeo-project.org (Last Access, April 2012).

[27] FAST (2012) http://demo.fast.morfeo-project.org (Last Access, April 2012).

[28] FAST GVS - Manual Part I (2009) http://www.youtube.com/watch?v=qFt2LBlxkwU (Last Access, April 2012).

[29] FAST GVS - Manual Part II (2009) http://www.youtube.com/watch?v=dpoRhnF8_1A (Last Access, April 2012).

[30] A. P. McAfee, "Will web services really transform collaboration," MIT Sloan Management Review, vol. 46, no. 2, pp. 78–84, 2005.

[31] J.C. Preciado, M. Linaje, S. Comai, S., F. Sanchez-Figueroa, "Designing Rich Internet Applications with Web Engineering Methodologies", in proceedings of the 9th IEEE International Workshop on Web Site Evolution (WSE 2007), pp.23-30, Paris, France, 5-6 October 2007.

[32] I. Garrigós, J. Gomez and G.-J. Houben, "Specification of personalization in web application design", in Inf. Softw. Technol. 52, 9, pp. 991-1010, 2010.

[33] M. D. McIlroy, "Mass produced software components," in Software Engineering, Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, October 1968, pp. 138–155.

[34] R. M. Balzer, "Imprecise program specification," Report ISI/RR-75-36, Information Sciences Institute, December 1975.

[35] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, and S. Wiedenbeck, The state of the art in end-user software engineering. Journal ACM Computing Surveys 43, 3, Article 21, April 2011.

[36] FIWARE EC FP7 Project, https://www.fiware.org/

[37] FIWARE Catalogue, http://catalogue.fiware.org/

[38] FIWARE Application Mashup Open Specifications, https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Apps.ServiceMashup. Last access: September 2015.

[39] FIWARE Application Mashup Generic Enabler (Wirecloud) http://catalogue.fiware.org/enablers/application-mashup-wirecloud

[40] D. Havlik, J. Soriano, C. Granell, S. Middleton, H. van der Schaaf, A. Berre, J. Pielorz, Future Internet enablers for VGI applications. In Proceedings of the 27th Conference On Environmental Informatics (ENVIROINFO), Hamburg; Sept. 2013.

[41] F. Ramparany, F. Galan-Marquez, J. Soriano, T. Elsaleh, Handling smart environment devices, data and services at the semantic level with the FI-WARE core platform, In Proceedings of the 2014 IEEE International Conference on Big Data (IEEE BigData 2014), Washington DC, USA, 27-30 Oct. 2014, pp. 14-20. DOI 10.1109/BigData.2014.7004417

[42] D. Lizcano, F. Alonso, J. Soriano, G. López, End-User Development Success Factors and their Application to Composite Web Development Environments, The Sixth International Conference on Systems (ICONS 2011), St. Maarten, The Netherlands Antilles, January 23-28, 2011.

[43] D. Lizcano. 2015, Statistical Survey of the End-User Development. Technical Report. http://apolo.ls.fi.upm.es/eud/

[44] A. Angeli, A. Battocchi, S. R. Chowdhury, C. Rodriguez, F. Daniel, and F. Casati, End-user requirements for wisdom-aware EUD. In Proceedings of the Third international conference on End-user development (IS-EUD'11), Maria Francesca Costabile, Yvonne Dittrich, Gerhard Fischer, and Antonio Piccinno (Eds.). Springer-Verlag, Berlin, Heidelberg, 245-250, 2011.

[45] F. Casati, F. Daniel, A. De Angeli, M. Imran, S. Soi, C. R. Wilkinson, M. Marchese, Developing Mashup Tools for End-Users: On the Importance of the Application Domain, International Journal of Next Generation Computing 3(2), 2012.

[46] D. Lizcano, F. Alonso, J. Soriano, G. López, A component- and connector-based approach for end-user composite web applications development, Journal of Systems and Software, Volume 94, August 2014, Pages 108-128, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2014.03.039.

[47] Centre Open Middleware Santander – UPM, http://www.centeropenmiddleware.com.