An Early Evaluation Method for Social Interactive Systems

Ines Di Loreto, Abdelkader Gouaich

AN EARLY EVALUATION METHOD FOR SOCIAL INTERACTIVE SYSTEMS

Ines Di Loreto, Abdelkader Gouaïch
LIRMM Research Report #RR-10016

ABSTRACT

Most evaluation methods address the perceived effectiveness of social aspect *after* user usage. On the contrary, the evaluation of 'potential sociability' in an application can help designers to anticipate several problems that can arise *before* starting the implementation. For this reason, this paper proposes an evaluation framework able to analyze social aspects and to give an 'early evaluation' of the designed application. In fact, the four dimensions that compose the framework (identity, space, persistence, and actions) can be used as 'indicators' in order to evaluate whenever the social system is able to facilitate the feeling of social presence. In particular, the presented paper will analyze the easiness of use of the method and its learnability.

Keywords

Social Web, Design Methods, Evaluation Methods, Social Presence

INTRODUCTION

Looking at current era we can notice that social aspects have to be considered as an essential component of the 'virtual' life of most current users. Current digital users - such as bloggers and gamers - take for granted social web features and expect them to be available into any application. For this reason, an application that fails in presenting at least familiar social features would be considered as a regression and may be rejected. Besides, the evaluation of social aspects in current software is performed a posteriori. On the contrary, we claim that an a priori evaluation can be useful. In fact, while users are able to evaluate the quality of their experience, for the most part they are not able to understand which feature/characteristic generates a poor performance. This does not mean that a user centered design approach (see e.g., [15]) is not useful when designing a social application. On the contrary, a deep analysis of users' needs is at the basis of any application development. However, between user centered design, and users' satisfaction measurement we want to add an intermediate layer. An early evaluation of the application design, in fact, can help designers to anticipate several problems that can arise *before* starting the implementation. Consequently, the early evaluation allows the designer to return on the phase of design to add missing elements if needed. This approach can help for example in restraining development cost. For the above-mentioned reasons, the authors proposed a framework based on four elements: identity, space, persistence, and actions. The framework was proved useful in building better social systems (see A2-A3. Missing citation for blind review-PHD thesis and internal report). In addition, because of its capability to analyze social aspects the framework can help in giving an early evaluation of the 'to be designed' application (see A1-Missing citation for blind review-paper accepted at the CSEDU 2010 conference). In fact, the four elements (and the behaviors they let emerge) can be used as 'markers' (or 'indicators') in order to evaluate if or not the designed systems is able to facilitate the feeling of social presence. This paper in particular, will focus on the learnability and easiness of use of the proposed method.

For this reason, following sections of this paper will be devoted to (i) define the concept of social presence (ii) present the framework and the method for early evaluation based on the framework (iii) show the results of an experiment where easiness of use and learnability were tested.

LEAVE BLANK THE LAST 2.5 cm (1') OF THE LEFT COLUMN ON THE FIRST PAGE FOR THE COPYRIGHT NOTICE.

PRESENCE AND SOCIAL PRESENCE IN SOCIAL ENVIRONMENTS

In his paper 'Measuring Presence in Virtual Environments: A Presence Questionnaire', Witmer [17] says that the effectiveness of virtual environments (VEs) has often been linked to the sense of presence reported by users of those VEs. For the author, presence is defined as the subjective experience of being in one place or environment, even when one is physically situated in another. In addition, presence is a normal awareness phenomenon that requires directed attention and is based in the interaction between sensory stimulation, environmental factors (that encourage involvement and enable immersion) and internal tendencies to become involved.

For example, in virtual worlds (such as Second Life) an active exploitation of our senses can create a psychological sense of presence, or, in other worlds, the illusion that 'I'm in the virtual world and not in my house' and, as a consequence, that 'I'm there with other people'[1].

What is interesting for this paper purpose is that we can extend the concept of presence to the concept of *social presence*. Social presence is, in fact, defined as the 'degree of salience of the other person in the interaction and the consequent salience (and perceived intimacy and immediacy) of the interpersonal relationships' [14]. In order to measure the potential presence awareness of our application several 'indicators' can be identified. Note that we are talking about indicators of 'potential sociability' before the implementation starts, not about indicators of sociability after system usage. The absence of these indicators of 'potential sociability' in a virtual environment is a signal of a not well-designed system, not able to support social interactions. As already said, being able to do such an evaluation at early stage (i.e., before starting implementation) has major advantages: it can simplify the work of designers (that can return on their design before development) and help to build better applications while reducing development costs. For example a designer can fix, add or remove features before starting to develop them, reducing in this case the development time.

DEFINING INDICATORS FOR SOCIAL PRESENCE

This section presents the core elements of the framework that can be used to evaluate social presence 'potential' in an application at early stage. The framework is based on four elements: identity, space, persistence, and actions. These elements are motivated by an empirical analysis of current and past social software and supported by major findings from psychology and sociology. Actually, these elements represent core features of any Social Interactive Systems (SIS) targeted towards young generations (see A2). Consequently, they represent interesting evaluation criteria in order to capture at early stage the potential presence awareness of the application being designed. Hereafter, the semantics of each element of the framework is described more in details.

Identity

Our point of view about Identity is the same as social psychology's approaches [7], which consider individual and social identity not as stable characteristics, but rather as a dynamic phenomenon [6]. In these approaches, the choice about what possible self to show is driven by strategic moves (e.g., what features are more relevant and effective for self-presentation) which participants can make within a particular situation. In describing everyday interactions, Goffman [5] distinguished between two ways of expressing information: information that is given and information that is given off. Information that is given is the conscious content of communication, the voluntary, symbolic actions that are mutually understood. For example, a person who describes their anger is giving information about their emotional state. In talking about their anger, however, the person also gives off information, through para-verbal characteristics such as tone, volume, the choice of words, and non-verbal cues. While information that is given is considered to be within the actor's control, information that is given off is perceived by the audience to be unintentionally communicated. A classical example of 'identity announcement' that has intentionally and unintentionally elements is avatar personalization. While we will not enter in detail here on its implications the avatar is a visual claim for personal expression that is constantly worked on. This continuous work reinforces the concept of presence and thus social presence. As another example of collateral information, we can use the explicit specification of a social network of acquaintance. While it is true that social networks are built via a series of invitations, usually members also have some control over the visibility of their network for others. This means that, for impression management, a user will show only networks he/she wants to show. For instance, some members can decide to make their social networks visible only to their direct acquaintances.

In this case, there is a 'given' information (the user chooses what to show about his/her identity), but also a 'given off' information (derived e.g., from the kind of groups a user showed/joined). From a design point of view, we can say that allowing both the kinds of identity representation becomes the starting point for a social evolving identity.

Space

If we look carefully, the language we use to describe our experience of the virtual environment is a reflection of an underlying conceptual metaphor: 'Cyberspace as Place' [10]. This means that we are transferring certain spatial characteristics from our real world experience over the virtual environment. The metaphor 'Cyberspace as Place' leads to a series of other metaphorical inferences: cyberspace is like the physical world, it can be 'zoned', trespassed upon, interfered with, and divided up into a series of small landholdings that are just like real world property holdings.

In this little presentation the term space was joined with the term place. In reality, for the good functioning of a SIS it is important to distinguish between the two terms. Actually, the literature about space and place is fairly massive and diverse. A converging definition of the difference between space and place does not exist, however in his book about urban spaces and places, Carmona [2] distinguishes among dimensions of an urban space. While space is divisible, place is not. Place is complex, inextricably multi-dimensional, lived, experienced, meaningful (with of course multi - meanings).

This means that while space is a well-defined topographical entity, place is the result of human inhabitation, (social) interaction, and the like. We are located in spaces, but we act and develop individual and social experiences in places. We claim that in order to design a social application, it is essential to allow by design the creation of public (at different levels) places for aggregation but also the creation of private places [16]. Besides, the lever of personalization can be used in order to allow the shift from spaces to places. Only taking possession of the space, and manipulating it to turn it in something we like, we can transform it in a place.

Persistence

As we have seen, in order to create a social identity in an online environment several elements are required. An additional element is persistence (of personal identity in the system). In a non-persistent world, it is not possible to have a history of actions and thus allow, for example, the creation of a reputation like in real life. Moreover, Danet [3] argued that synchronicity is associated with 'flow experiences', a state of total absorption and a lack of awareness of time passing. This idea of synchronicity is linked to the idea of temporality, a linear procession of past, present, future. This particular nuance (synchronicity as process) is very interesting if we think that interaction with media and media perception is changed. In fact, advances in technology and the speed of network connections are blurring distinctions between synchronous and asynchronous communications [8]. Synchronous and asynchronous communications are thus processes that happen during time. The idea of communication as a process is very consistent with the idea of persistence and is another element supporting social awareness.

Actions

In this part, we discuss physical and psychological mechanisms that regulate human actions in order to understand why the action element has to be considered as a pillar in the design of social software. The first theory we want to describe is the so-called 'thinking through doing'. This theory describes how thought (mind) and action (body) are deeply integrated and how they co-produce learning and reasoning [9]. Jean Piaget [13] postulated that cognitive structuring requires both physical and mental activity. In a very basic sense, humans learn about the world and its properties by interacting within it. As a second support, we can cite embodied cognition. Theories and research of embodied cognition regard bodily activity as being essential to understanding human cognition [12]. While these theories address cognition through action in physical environments, they also have important implications for designing interactive systems. In fact, body engagement with virtual environments constitutes an important aspect of cognitive work. For example, one might expect that the predominant task in Tetris is piece movement with the pragmatic effect of aligning the piece with the optimal available space. However, contrary to intuitions, the proportion of shape rotations later undone by backtracking increases (not decreases) with increasing Tetris-playing skill level. In fact, players manipulate pieces to understand how different options would work [11].

To summarize, because an action is always an action-over-something, the kind of interaction spaces and objects we create in a Social System will influence which cognitive work the user will do over the system.

THE OVERALL FRAMEWORK

While we presented the four elements in a separate way, their usefulness in the construction and evaluation of social environments is mostly linked to the interaction between these elements.

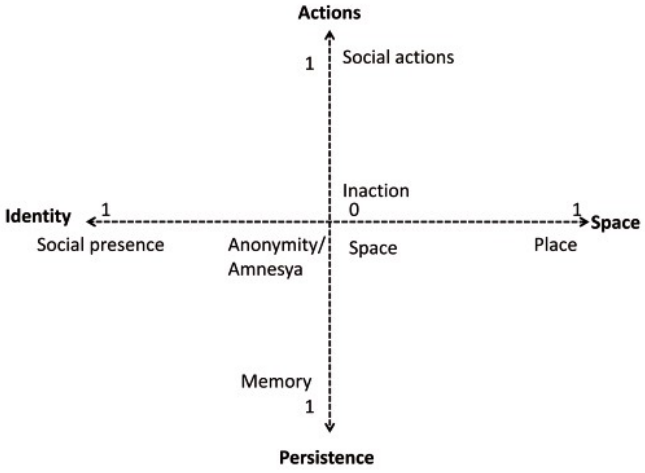

Fig. 1 A graphical representation of the four elements through axis.

In a way, each element can be thought of as a line (an axis) that starts from the absence of the element to the fulfillment of its presence for a Social Interactive System. For example, for the concept of identity its total absence is anonymity while its fulfillment is social presence (with intermediate points such as personal identity construction).

For the concept of space its total absence is topographical space while its fulfillment is social places (with intermediate points such as third places and personal places). For the concept of persistence its total absence is system 'amnesia' while its fulfillment is memory (with intermediate points linked more or less to the concept of persistence). Finally, for the concept of action its total absence is the obstruction of action (i.e., my user can only look at my application) while its fulfillment is social actions (with intermediate points such as public personal actions and the like). Fig.1 shows the above-described axis graphically. This way a designer can create an 'Expected Profile' for an application. For example, if he/she decides

that his to be developed application has to have a high level of self-presentation elements (an avatar, a profile, and so on) he/she will give a high value for the identity axis. Same thing happens for the persistence axis. For example, a social network based on micro actions such as Facebook, does not require the same level of persistence as a virtual world such as Second Life. In the first case the persistence axis will have a medium value, in the second a high value. And so on[1].

[1] Note that this explanation is an oversimplification. In our work (A3-missing for blind review) we described a very detailed set of elements each designer has to take into account in order to design different social systems.

	Identity	Space	Time	Actions
Identity	--	Personalization (Self expression)	Persistence	Performance
Space	Personalization (Self expression)	--	Persistence	Personalization (Self expression)
Time	Personalization (Memory of changes)	Personalization (Memory of changes)	--	Performances through time
Actions	Reputation	Environment appropriation (personalization)	Persistence	--

Tab.1 The relationship between the four factors

However, the total framework is not simply a list of elements (i.e., its application does not mean to put one after the other the four elements in your system) but it's created through the delicate balancing between them. Table 1 shows in a simplified way the relationship between the four elements of the framework. Actually, it is up to the designer to choose which element of the framework to stress or not during the creation of a dynamic experience such as in a social application. Only once the 'Expected Profile' of the application is decided, the designer chooses which features add to the systems.

A METHOD FOR EVALUATING AN EXISTING DESIGN

While the above-described framework can also be used in the phase of design of a social application, for this paper we want to focus on its application for evaluation purposes.

In fact, the framework can be helpful to evaluate the 'potential sociability' of an application. For this reason, in this section we propose an 'Early Evaluation Method' (from here on EEVa method) based on the framework. If the use of all the four elements is fundamental to build 'good' (successful from the social point of view) applications, a way to measure their absence/presence at early stages can help to avoid developing 'unsuccessful' systems (i.e., systems that are destined to have a negative impact on sociability).

Normally, software development starts with a design document. Using the design document, engineers decide a set of software requirements. To accomplish this, they employ use cases and other tools of analysis. After that, they design the software (and finally someone develops it).

The major problem of this approach is that the starting design document provides information about the application from the perspective of the designer (who could be a computer scientist but also a pedagogue, and the like). Let's use a game design as an example. The game designer seeks to create a game with a map editor, a character editor, several levels, and a complex world. This means that the game design document provides information about the game as an artistic entity. However, the requirements that are drawn from this document constitute the first important step in transforming the vision of the game into technical specifications.

The Idea of Stripes

For this reason, after the design phase each developer faces the problem of 'how to translate the design in natural language into software requirements'. In order to overcome this difficulty, an interesting approach is the use of the 'stripes' concept.

In Flynt's definition, a stripe is 'a set of functionalities embodied in a single component of the system.' [4]. More specifically, a stripe embodies a subset of the functionalities described in the requirements document (for example the GUI - Graphical User Interface - can be a stripe). Generally, the first stripe consists of the most general system features only, such as the framework of the application. With each successive stripe, the features addressed become more refined. The level of detail and complexity grows with each stripe, but because the detail and complexity are layered, at no point does complexity become overwhelming. While it's true that the stripes approach calls for designing all stripes before beginning software construction, it also involves an approach that is iterative and incremental. In fact, after the creation of the stripes, priorities are given to their development. In addition, priorities can also be given to the setting of features they embody. This kind of management creates an incremental approach at both levels: the single stripe and the whole system.

A Method for Translating Designs

The interesting part of this approach (i.e., the part that suggested a possible link with the framework described in this paper) is its incredible use of the concept of 'chains of actions'. The problem with the same approach is that it does not give any (more or less) 'formal' way to translate the natural language design into stripes. For this reason, the need arises to develop a method for 'translating' the design into stripes. In order to make this 'translation', an iterative phase of pre-design was inserted between the concept

development and the classical computer scientist design phase. The idea behind this insertion is to use the pre-design phase to translate the design into requirements in the

Sub-Stripe N° (action)	Stripe description	Who (who makes the action)	When (persistent?)	Interaction Space	Non functional(adjectives)	Issues	IdW	SpW	TimeW	ActW
0(Example)	Example: the player can change his face: shape, eyes, etc	Who (the actor of the action): the player	When (stable effect on the world or not; session duration, time duration): stable ('till next change)	What (on what): the avatar	(i.e. for the graphic designer only, for the sound creator, and the like)	What do you mean with?? Fill this field if you don't understand something	1	0	1	1

Fig. 2 The schema for the sub-stripes creation

following way. The designer takes the natural language specifications; he/she analyzes them through a schema based on the framework and finds the actions that characterize the design. Each of these actions can be defined as a 'sub-stripe' (i.e., a part of the total stripe). For example, the path to follow in order to complete a quest in a game can be seen as a stripe (find a better sword, find the monster, kill the monster, acquire experience), while the fact that it is possible, for example, to kill monsters in order to fulfill your quest, is a sub-stripe. Once the designer has defined all the sub-stripes for the design he/she joins them into a set of chained actions that will result in the stripes (for example the quest we mentioned above). The analysis of each stripe will allow the designer to de-construct them into features (note again that features can be common to different stripes) and components. In our previous example, the system that manages experience every time you kill a monster can be seen as a feature. Obviously, this feature can be shared by other stripes.

A path for the Translation

In order to follow the path described, a detailed list of actions for helping the 'translator' in his deconstruction was created (see also Fig. 2).

As a first step the 'translator' does a very simple thing: he/she takes the design and highlights actions (i.e., verbs) in natural language. For each action he/she then defines the elements that impact on the framework using the field of the table. The 'Who', 'Interaction space', and 'When' elements, for example, help to define the persistence (or not) of the action and if the action concerns identity or space (who is acting: the user, the system and the like, and 'over what' he is acting: the avatar, the system the Graphical User Interface and the like). The application of this method is linear (in fact the 'translator' analyzes the document paragraph after paragraph). The result of the application is a set of very detailed sub-stripes. Note that the issue field was inserted in order to facilitate the 'feedback cycle' on the design. If something is not clear to the translator, he/she simply opens an 'issue', i.e., he/she ask designers to explain the element better. At the end of the first analysis the translator sends all the issues to designers and when he/she receives the answers he/she can decide for example to modify actors, etc. for the related sub-stripe.

At this point of the explanation the need arises to clarify two subjects. Firstly, In Flynt's textbook there are other described frameworks (i.e., function, object-oriented, patterned). The decision to use the Stripes method is linked, as said before, to the fact that this kind of development is compliant with the concept of activity (action) present in the framework. Secondly, Flynt does not offer a method for translating the design into requirements. The adoption of an iterative pre-design phase, the creation of the translation method, and the early evaluation method that will be described in following sections are an addition made by the authors.

Evaluating the Design through the Framework

As said, the reason for this paper is to evaluate easiness of use and learnability of an 'Early Evaluation Method' (EEVa Method) based on the framework. This EEVa method implies the possibility to evaluate the design before starting the development. For example, it may happen that a designer wants to create a design that is divided equally between all the four elements. However, the application of the EEVa method to his/her design could show that the design is missing something (e.g., in identity features). At this point the designer is still in time to add missing features before implementation starts.

In particular, sub-stripes can be used in order to make the 'Early Evaluation' of the designed application. In fact, filling in the schema for each sub-stripe implies 'explicit' actors/agents (of the action) and objects.

First of all only the sub-stripes that fall under the label of the four elements of our framework are analyzed. For example, the fact that the system has a splash screen can be counted as an action but not as an action that impact on the system. Then, we can provide weights, ranging from 0 to 1 to each sub-stripe. The translator will give 1 to the ID field if and only if the element impacts on the system, otherwise he/she will give 0. The same will be done for the other elements.

Once the weights are given, the final action is simply to sum up all the weight present in the different columns.

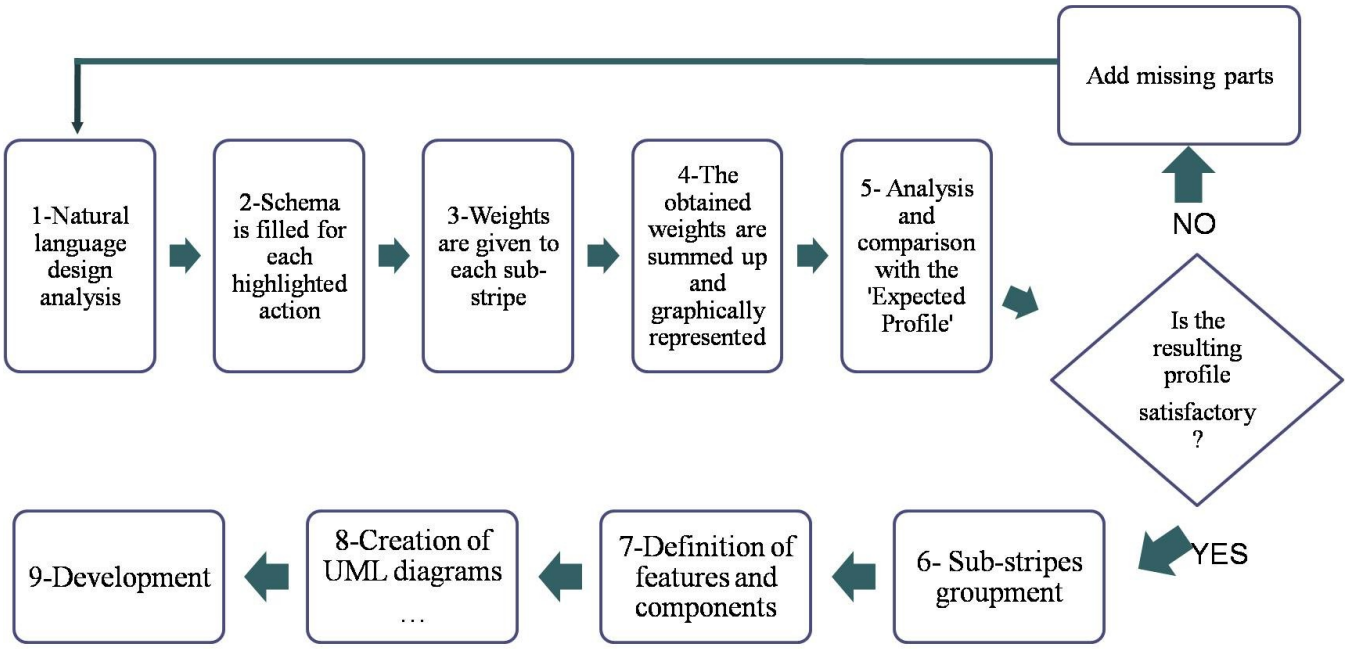

Fig. 3 The overall steps for the EEVa Method

The four obtained weights can then be graphically represented through the four axes.

To summarize, the above-described process for the 'EEVa Method' can be described as (see Fig. 3):

Step 1: The translator analyses the natural language design in a linear way and underlines all the actions (verbs). If something is not clear, the translator can pose questions using the issue field.

Step 2: for each highlighted action, the translator fills in the schema in Fig. 2. In particular for each action he/she answers: Who makes the action? It's a persistent action? What does the action impact on ?

Step 3: The translator gives weight to each sub-stripe, based on his answers.

Step 4: The obtained weights are summed up and graphically represented.

Step 5: The designer analyses the graphic and compares it to the 'Expected Profile'. Note that if there is no 'Expected Profile' this step of the process requires the presence of an expert of the framework and its family resemblance.

If the analysis it's not compliant with the desired result, the design has to be reworked and then the process re-applied for the added/modified parts starting from Step 1. On the contrary, if the analysis is considered suitable, the designer can continue his work.

Step 6: Once all the sub-stripes have been obtained the designer regroups them into stripes, using the Who and What fields as guidelines.

Step 7: For each stripe the designer defines the linked features, and then the derivatives of the components.

Step 8: Using the defined objects the designer creates UML diagrams (dependencies, use-cases, etc.).

Step 9: Development starts.

Note that the part strictly linked to the framework is only the first cycle (from Step 1 to Step 5). The other steps are a consequence of the stripes approach.

THE EXPERIMENT

In order to: (i) check practicality of the proposed methods when building an actual complex social system; (ii) evaluate qualitatively the benefits of such methodology, an experiment involving 'real world' designers and developers was carried out. Results of the study are reported hereafter.

Subjects

The participants of the experiments were 26 computer scientists from France and Italy. The gender distribution of participants was 22 (85%) males and 4 (15%) females, with an average age of 27 years. 15 of them were master students in computer science, while the rest of the participants were professionals working in the sector.

Procedure and Materials

Two 3h ad hoc sessions conducted by the same person (one for the French native speakers, another for the Italian ones) were held to introduce participants to the method.

At the end of the session each of the participants received a file summarizing the framework, the method, and the 'Expected Profile' concept.

Participants were then asked to apply the method to a particular natural language design (a social network for friends use called MyMellon), and to follow a defined procedure (see next paragraphs). At this moment they were also given an 'Expected Profile' for this application.

At the end of the experiment participants were asked to fill in a survey and another discussion session was held.

In particular, participants were given the following guidelines:

1) Analyze the design in order to find actions:

1a) highlight each action in the design

2) Fill in the fields of the schema (sheet 1: Substripes) FOR EACH action, answering:

2a) who does the action? (who field)

2b) is it a persistent action? (when field)

2c) the action impacts on what? (what field)

3) Is the action non functional? (is it for the coder? If yes, it's a functional action, otherwise if it's for the sound manager or the graphics man it's a non functional action)

N.B. If something it is not clear, use the issue field to pose questions to designers.

4) Give weights (0/1) to each sub-stripe. Give 1 to the ID field if and only if the element impacts on the system, otherwise give 0. Do the same for all the other elements.

5) Evaluate the resulting graph: Do you think that the result is compliant with the expected profile?

Do you think that the design has to be improved?

6) Regroup the sub-stripes into stripes using the who and what field. (fill in sheet 2: stripes) For example, regroup all the action that impacts on the Gui, on the avatar and so on.

7) For each regrouped stripe define first features, then components

8) Create all the diagrams you think could be useful for real development.

No time constraints were given and the participants were not asked to develop the application. On the contrary, participants were asked to deliver the final excel sheets with the elaborated sub-stripes and stripes, as well as the UML diagrams.

Results and Basic Discussion

Before starting an in depth analysis of the results it's worth noting that each of the participants naturally performed the translation process in two steps. Practically all of them (24 out of 26), in fact, sent issues to the designer via e-mail. Only when they had their answers did they resume the process.

After this introduction, from the collected data regarding ease of use and perceived utility some general considerations can be drawn:

Ease of use (based on the survey filled by participants) rated 2,5 (scale 0-4). On average students rated it lower than professionals. However, lower marks were given by two professionals. The reason for this mark (in participants' words) is that they are used to a non linear-analysis of the design. This means that after reading specifics, they immediately produce UML or E-R diagrams without intermediate steps. For them the way they were asked to think was too detached from the way they were used to thinking.

Regarding perceived usefulness of the method, 75% (20) of participants rated it as Very Useful, 25% Quite Useful. In addition, the two people who rated it as very difficult to use, rated it Quite Useful. One of the reasons for noting this was the perceived importance of analyzing the design as carefully as possible before programming in order to avoid errors during the software development. One of the more 'enthusiastic' stated: 'It allows you to clarify the requirements of your application in your head before committing to designing the architecture. If necessary you could then fix, add or remove features depending on your objectives.'

97% of participants considered the time used for applying the EEVa Method the right amount of time for an early evaluation. The same number of participants asserted that the method forced them to look at the design more carefully than in normal cases.

An interesting point to note (although not strictly linked with the framework) is that while students regrouped the sub-stripes practically in the same way, professionals were very creative. While this is not relevant for the EEVa method evaluation, this is an indicator that the remaining part of the stripe method can also be applied in a flexible way. However, as has been underlined by participants, the method does not have any specific tool or guideline that prevents one from creating a badly formed stripe. Thus a complete understanding of the UML design process is necessary in order to fully appreciate this method.

Strictly speaking about the evaluation of the EEVa Method, all the participants found it very easy to evaluate the application (results are the same for all the groups).

Finally, some words about the issue field. The' issue' field was considered by everyone to be useful for interacting with the people who worked on the conceptual phase. It was also underlined that it opens up the method to a collaborative approach.

GENERAL DISCUSSION

Different kinds of considerations are necessary after the above reported data.

First of all the perceived ease of the EEVa was also due to the semiautomatic process created through the excel file. In fact, once the participants gave the weights, the excel file automatically calculated the sum and generated the graph for the evaluation. All the participants had to do was to compare the result with the profile they were supplied with.

Not surprisingly at all (for a computer scientist sample), 80% of the surveys had a comment asking for an automatic language analyzer able to extract subjects, etc. from the design, or at least a pre-filled pull down menu (avatar, system, and the like).

While we are not against a pre-filled pull down menu (for example the 'When' field caused questions like: 'Can I write OnClick in here?'), we do not think that relying on an automatic language analyzer can really be helpful. In fact, a natural language phrase can be modified by the translator during the process in order to fulfill his needs. For example a phrase such as 'an animated intro shows...' can also be interpreted by the translator as 'the system has an animated intro that shows...'. Moreover, a completely automated process (i.e., a process that automatically gives weights based on the terms used in the fields) takes away all the translator control (i.e., all the human control). To summarize, we think that providing an automatic 'finder' for verbs can be useful (the experiment highlighted that not all the participants had the same familiarity with language analysis). In the same way, a drop down lists, maybe pre-filled by the same translator, can aid to fill in in a simplified way also the weight parts. On the contrary, we are against a totally automated process.

More important than the automatic approach to the analysis, is the knowledge about the consequence of the absence of one element of the framework. In this experiment, participants had to compare the resulting profile with a given one and decide to improve the design or not. In order to do this, most of them were not willingly to 'extract' the information about the framework from the supplied files and relied on their memory (i.e., on things learned during the first ‘live’ session). It is our opinion that this kind of approach can be a limitation. The expertise on the framework for an 'evaluator' is very important. In fact, only an 'expert' can answer questions such as: What happens in the case of an application that is between a general social network and a virtual world? Which profile is the best? And so on. While detailed guidelines for specific class of applications and fully structured examples can surely reduce the steepness of the method's learning curve, only an in depth knowledge of the framework can create a good Social Interactive Systems analyzer.

From the practical point of view, one evident limitation of this method is the fact that, while it can be applied to whatever natural language design you want (from games, to social networks, to accounting software), you need a natural language design. Now, in real life most small projects start with an idea that is re-worked while developing (i.e., parts are added time to time). In this case the EEVa delays the developing starting point too much. In fact, the method schedules for cycling the EEVa each time you add consistent parts to the natural language design. If the added parts are too little the number of cycles become overwhelming. To summarize, this method is better applied to complex software development which requires written requirements. Nevertheless, the need for an EEVa able to analyze the 'social potentiality' of the application still remains. Insofar as we know there are no methods on this subject.

The last comment on this method regards the 'public'. During the experiment we saw the emergence of three sub-sets of population. One can be called 'the students'. They had to be guided more during the application of the method and they produced more uniform groups of stripes. The other one can be called 'conservative professionals' – for them applying the EEVa method was too far from the way they are used to developing software. Note that they were not the older professionals involved in the experiment, so this is not a matter of age. Finally there is a population of 'flexible professionals' who added their own expertise to the experiment. Actually, this is a 'method for everyone' (especially if other levels of automation are added), but not a method useful in the same way for everyone. ‘Flexible professionals' are the best candidates for the 'position' of 'profile analyzers'. During the experiment they were proactive (every single one of them read the written guidelines on the framework) when analyzing the resulting profile. Students, on the contrary, really need a set of examples to rely on, and they were not able to explain why the profile needed improvement. While this is also a way to apply the method, it's certain that the other one leads to a deeper understanding of the resulting profile consequences in terms of 'potential sociability'. In fact, while this paper has only shown the easiness of use and the learnability of the method, the real purpose of the proposed ‘Early Evaluation Method’ is to evaluate ‘potential sociability’ of an application in order to avoid time (and money) consuming developments. In order to be able to do this, the EEVa Method is useful, but the knowledge of the framework is essential.

CONCLUSIONS

This paper started from the assumption that an evaluation of 'potential sociability' in an application can help designers to anticipate several problems that can arise *before* starting the implementation. For this reason, the paper proposed an evaluation framework able to analyze social aspects and to give an 'early evaluation' of the designed application. In fact, the four elements that compose the framework (identity, space, persistence, and actions) can be used as 'indicators' in order to evaluate if or not the social systems is able to facilitate the feeling of social presence.

In particular, this paper analyzed easiness of use and learnability of the 'Early Evaluation Method' based on the framework. The experiment carried out demonstrated that the method is easy to apply and requires a perceived normal amount of time for its application. The method has also shown two possible ways of application: one more 'passive' and another more 'active'. In the first one, the choice of whether or not to improve the application is based on comparing the resulting profile with the 'Expected Profile' of the application. In the other the choice is a mix between the knowledge of the framework and the 'Expected Profile'. This second attitude is very important if the 'evaluator' also has to be a 'designer' of a Social Interactive System.

REFERENCES

1. Biocca, F. The cyborg's dilemma: Progressive embodiment in virtual environments. *Journal of Computer-Mediated Communication 3*, 2, (1997).

2. Carmona, M., Heath, T., Oc, T. and Tiesdell, S. *Public places-urban spaces: the dimensions of urban design.* Architectural Press, 2002.

3. Danet, B., Ruedenberg-Wright, L., and Rosenbaum- Tamari, Y. Hmmm... where's that smoke coming from? writing, play and performance on internet relay chat. *Journal of Computer-Mediated Communication, volume 2,* (1997).

4. Flynt, J. P. *Software Engineering for Game Developers*. Software Engineering Series. Thomson, 2004.

5. Goffman, E. *The presentation of self in everyday life*. Doubleday, 1959.

6. Harré, R. and Langenhove, L. V. Varieties of positioning. *Journal for the Theory of Social Behaviour, 21*, 4, (1991), 393–407.

7. Hogg, M. Social identity and group cohesiveness, in J. Turner, M. Hogg, P. Oakes., S. Reicher, & M. Wetherell (eds.), *Rediscovering the social group: A self-categorization theory*. Oxford: Blackwell, (1987) 89-116.

8. Joinson, A. N. *Understanding the Psychology of Internet Behaviour: Virtual Worlds, Real Lives.* Palgrave Macmillan, 2003.

9. Klemmer, S. R. and Hartmann, B. How bodies matter: Five themes for interaction design. In *Proceedings of Design of Interactive Systems 74* (2006), 140–149.

10. Lakoff, G. and Turner, M. *Categories and Analogies 3*, (1988). University Of Chicago Press.

11. Maglio, P., & Kirsh, D. Epistemic action increases with skill. In LEA (ed.), *Proceedings of cognitive science society* (1996).

12. Pecher, D. and Zwaan, R. A. *Grounding Cognition: The Role of Perception and Action in Memory, Language, and Thinking.* Cambridge University Press, 2005.

13. Piaget, J. *The origins of intelligence in children*. International University Press, 1952.

14. Short, J., Williams, E., and Christie, B. *The Social Psychology of Telecommunications*. John Wiley and Sons Ltd, 1976.

15. Vredenburg, K., Isensee, S., and Righi, C. *User-Centered Design: An Integrated Approach*. Prentice Hall PTR, 2001.

16. Wenger, E., Mcdermott, R., and Snyder, W. M. *Cultivating Communities of Practice*. Harvard Business School Press, Boston, 2002.

17. Witmer, B. G. and Singer, M. J. Measuring presence in virtual environments: A presence questionnaire. *Presence, 7*(1998), 225–240.

18. A1-Missing citation for blind review.

19. A2-Missing citation for blind review.