2

Human-Machine Interaction and Agility in the Process of Developing Usable Software: A Client-User Oriented Synergy

Benigni Gladys[1] and Gervasi Osvaldo[2]
[1]*University of Oriente*
[2]*University of Perugia*
[1]*Venezuela*
[2]*Italy*

1. Introduction

The human-machine interaction model, in its simplest meaning, describes the issues related to an individual who interacts with a machine in a given context. At the level of information processing, each input and output, and each related process is modelled on the behaviour of the machine and of the human being. It is worth noticing that the inputs of a person (defined as the actors) are her/his sensorial organs (e.g. sight, hearing, smell, etc.) and the outputs (defined as the effectors) are the fingers and the voice. On the other hand, the inputs of a machine are represented by the interactive devices (focused on controlling the execution of the program) such as the keyboard and the mouse, which are classified as non-immersive virtual reality devices, while the outputs are represented by the presentation devices (focused on the display of the information) such as the screen, the sounds and the alarms.

These inputs originate from the interface of a computer application, which provides a series of actions (e.g. a click) in which the main actor in the human-machine interaction is *the user*. We have to take into account that in the production of web or desktop applications the final product must conform to the highest standard in quality considering the high competitiveness of the information society. Nowadays the progress made in Software Engineering, Usability and Object-Oriented Software Engineering has often made available usable agile products, thanks also to the invaluable contributions of gurus like Nielsen and Norman. Agile usability is a recent concept developed by Nielsen & Norman (dateless), and the term *agile* is a paradigm in software production.

The term *agile development process* substitutes the classical waterfall model in the production of software as stated by Grezzi (2004) who considers the final user as the key element for success in the design of any software application. We fully agree with the assertion of the authors of the Manifesto for Agile Software Development: *the contribution of people is more relevant than the processes or the technology used*. Ultimately we feel that a synergy or symbiosis among different disciplines characterized by usability engineering, agility, software quality and the human factor exists.

The aim of the present paper is to emphasize the importance of agile software development for releasing software products based on the Human Machine Interaction paradigms and centred on user needs and expectations. The agile usability and the object-oriented software engineering perspectives are at the centre of the process to achieve agile methods and to release products centred on users. To reach this goal, the model based on the Human-Machine Interaction principles is proposed, and conceptual models related to each one of the actors (user-client, designer-programmer) who take part in the different stages of software development are presented.

To show the importance of agile usability, the ISO standards quality and the agile method are used to propose some development methodology (USABAGILE Web, SCRUM, among others) which would be the starting point of the process of delivering high quality products centred on the needs and expectations of the *user*, the entity on which all activities are orientated.

2. Human machine interaction and its development through technical Interaction

The Human Machine Interaction (HMI) is related to the study of the interaction between the human and the machine (represented in our case by the computer) and the tasks the users are routinely carrying out using software interfaces. The important aspect of HMI is the knowledge of how machines and humans interact in order to carry out the users tasks in their context.

The HMI found the solution for several problems in computational science, by combining several disciplines like social and physical sciences, engineering and art [Martinez 2007]. The HMI takes advantage of the advances in computational science, psychology, mathematics, graphic arts, sociology, artificial intelligence, linguistics, anthropology and ergonomics. Therefore, a software developer has to take into account the four main components of a human-machine system:

a. the user;
b. the machine and its system;
c. the task;
d. the environment.

The main issue is to comprehend the user in the environment in which she/he carries out her/his tasks, in order to design applications suited to her/his needs. Furthermore, the implemented software has to respond to a request specified through an input device and it has to receive all the necessary information through a device (e.g.: a computer). It is hard to know when the interfaces were introduced, since the human being has interacted with machines from the prehistoric age. What is really clear is that interfaces evolved in order to make human life more comfortable and easy, according to a design centred on human being.

The interfaces' evolution depends on the devices, the operating systems, the programming languages and the ways users approach computers and mobile devices. In particular the most important driving factor of the interfaces' evolution was the advent of operating systems based on a Graphical User Interface (GUI), combined with the evolution of Smart

Phones and mobile devices, which make information available in a ubiquitous and very intuitive way. The future evolution of computer technologies will be driven by the concept of ubiquitous computing and natural languages and interfaces.

3. The human processor as a reference point in the processing of information: user conceptual model, designer conceptual model and programmer conceptual model

To study the human factor designing interactive systems we take into account the cognitive psychology, which according to the Manning definition [Manning 1992] is *the study of those mental processes which make easily possible the recognition of familiar objects, known people, the management of our surrounding world, including the skills of reading, writing, programming, plans execution, thinking, decision-making and memorizing what one has learnt.*

To understand and to take advantage of the human-machine interaction concepts, it is necessary to understand the human memory and cognitive processes. The model of the human processor is shown in Figure 1. This model has been included in the model of multiple memories to achieve a better comprehension of the global process of learning. In the field of Human Machine Interaction there are various research areas which are devoted to the understanding of the various processors, in order to define principles and guidelines for designing good and efficient interfaces.

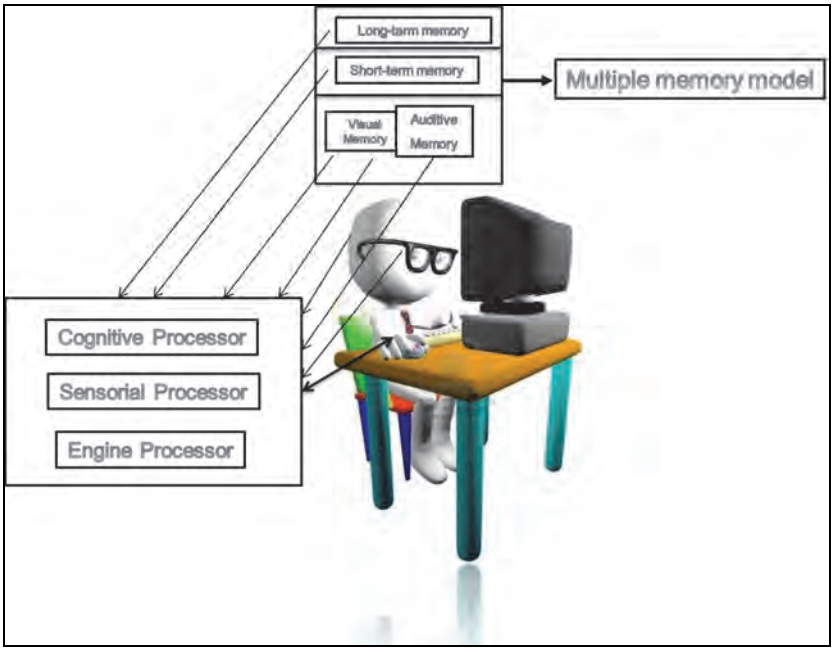

Fig. 1. Model of the Human Processor.

The learning research area involves the comprehension of the human processor model, under the assumption that taking advantage of the capabilities of gathering information, we

are facilitating the process of a more efficient and faster learning process. The Engine Processor is responsible for efficiently managing all interactive mechanisms suitable for a given interface affecting the learning process. The Sensorial Processor captures the information after having received some stimulus from the environment through the human senses, transforms it in a concept and then transfers the concept to the Cognitive Processor.

The Sensorial Processor uses a memory area (buffer) for each sensorial organ (sight, hearing, smell, taste and touch). Finally, the Cognitive Processor is of utmost importance in the learning process. We can consider a learning process as fast in all cases in which we achieve a rapid transfer of information between the short and the long term memory, assuming that the transfer between the sensorial memory and the short term memory has been done properly.

Considering that the model of the human processor is able to receive and process information in response to a sensorial stimulus, we have to know the users and their contexts in order to design and implement suitable interfaces centred on users. For this reason the main aim of the involved parties in the development of an agile and usable product shown in Figure 2 is to cooperate, in order to achieve the goal of releasing agile and usable products.

Fig. 2. Aims of the actors of usable agile products development.

When modelling the application, one has to consider the user perspective together with the designer and the programmer perspectives. It is important to emphasize that, even though the user is the key element on which the application has to be focused, being the buyers of the product and the final users of the interface for performing various tasks (user perspective), other actors are crucial too. In fact, the designer outlines usable interfaces, providing the user with all the necessary facilities, the comfort and the ability to complete a given set of tasks easily (designer perspective). Finally, the programmer implements the application following all specifications provided by the designer (programmer perspective).

In Figure 3, considering the relative importance of the actors described above, we can show the metaphor corresponding to the mental models of each of them playing a role in the process of building a user centred application.

3.1 The user model

To be able to create the user model it is fundamental to know the user's experience, her/his knowledge and expectations. We can become acquainted with such issues by carrying out usability tests (observing, polling, asking, etc). Such tests have to be carried out with the final users of the product.

If the interface is implemented in the wrong way, the user may have a strange behaviour in the future, while trying to counteract the weakness of the application.

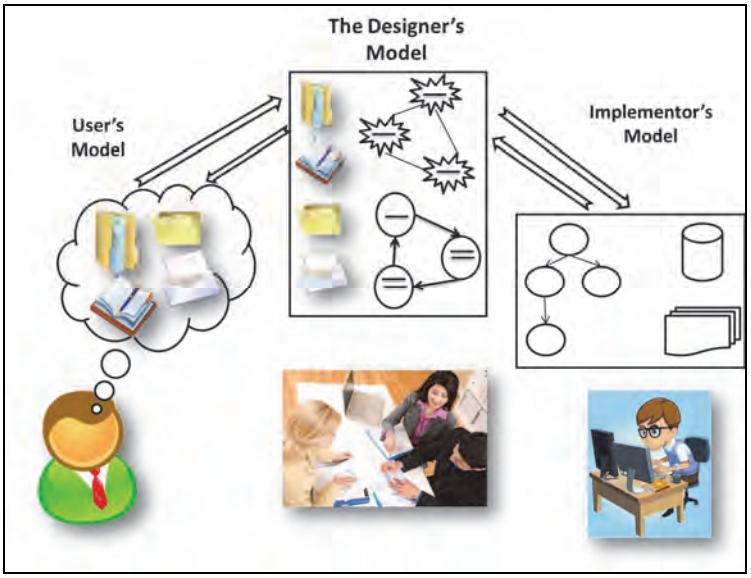

Fig. 3. Mental models from the actors.

3.2 The programmer model

The programmer knows which are the software developing platforms, the operating systems, the developing tools, the programming language and the specifications required to deliver the application. Such knowledge does not necessarily allows the programmer to produce suitable interfaces. The programmer implements the interfaces according to the specifications received from the designer model.

3.3 The designer model

The designer has to figure out what the user will perceive and see using the application. The designer translates into the computer domain the comprehension and the analysis of the User Model. In Figure 3 the interaction of the designers with the other actors is stressed, since if we want to plan properly the correct model of a given software product, we have to consider that: a) the designers have their own system model; b) the image of the system is implemented according to a given plan; c) the User Model is built through the interaction with the system.

The designer hopes that the User Model fits her/his own model. In any case, the communication is performed through the system. The system has to reflect a model of a clear and consistent design between the User and the Designer Models.

4. Object-oriented software engineering and its interrelationship with usability engineering

In section 2 we stressed the importance of user-centred design and once again we refer to this concept in order to link Object-Oriented Software engineering (OOSI) with Usability Engineering (UE); to this end it is relevant to mention the characteristics of the User-centred Design (UCD) based on the standard ISO 13407, whose role is: (a) to actively involve users and clearly understand the requirements of the user and the task; (b) to define an appropriate distribution of responsibilities between the users and technology; (c) to highlight the iteration of design solutions and multidisciplinary design.

We need to be involved in each phase of the software development to understand and define the context of use, the tasks and the way in which users work and how they work with the developed product. Let us remember that the success of the development and subsequent use of the software can be achieved involving the users and the customers in each stage of the development process. The user-centred design leads to an interactive design, where in particular the feedback offered by users and customers is a source of fundamental information to achieve the goal. Martínez la Teja said: (Martínez la Teja, 2007)

> *It requires combining a variety of skills and knowledge depending on the nature of the system to be developed, which is why the multidisciplinary team may include members of the management, experts on the application, end users, system designers, experts in marketing, graphic designers, specialists in human factors and training staff. It is possible that one person represents several of these areas, but something important to consider is that the designer can never represent the user, unless the design is developed for her/his personal use.*

For this reason today in each of the phases of the development of a software product, there is a pool of users, customers, designers and programmers who interact and obtain an incremental and iterative design of the application to be launched into the market. Currently all actors work jointly in the various phases of the cycle of traditional development systems (starting from the most known and used: the "waterfall model"), combining the Software Engineering principles updated with the Object Oriented ones, with their different phases, tools, techniques, and models available through the Unified Modeling Language (UML). Furthermore, the software development has to be carried out in parallel with the life cycle of the Usability Engineering.

4.1 In the software development life cycle

One of the most basic structured models (Lorés & Granollers, s.d.), serving as a block construction for other models of systems development life cycle, is the waterfall model (see Figure 4). Currently, the waterfall model is being replaced by iterative and incremental models associated with the latest technology of the object oriented programming, although the waterfall model is still the basis of all implemented models (see for instance the USABAGILE Web model described in section 7).

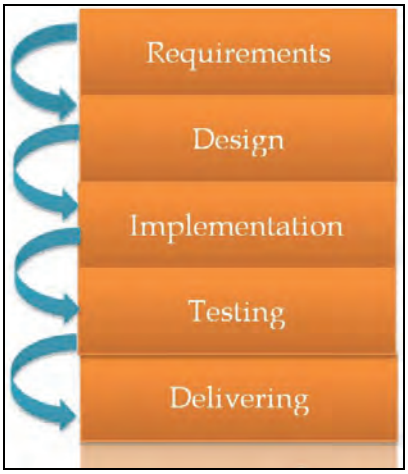

Fig. 4. Waterfall model.

As you can infer from Figure 4, the waterfall model starts from the collection of information culminating with the release of the software product. No feedback is present between the various stages, which was one of the failures of the method.

4.2 Software engineering

Zavala's definition of a software product (Zavala, 2000), *"a software product is a product designed for a user"*, is useful to understand how Software Engineering represents an important systematic approach to the development, operation, maintenance and removal of software, which allows cost-effective and usable software products to be released. The process of software engineering is defined as a set of stages partially ordered with the intention of achieving a goal, in this case, a software quality (Jacobson, 1998) output. In the process of software development the needs of the user are translated into software requirements, transformed into design requirements and then the design is implemented in a code. Finally, the code is tested, documented and certified for operational use. Specifically, it defines who is doing what, when to do it, and how to achieve a certain objective (Jacobson, op. cit.). The process of software development requires a set of concepts, a methodology and its own language. This process is also called the *software life cycle*, which is composed by four major phases: design, development, construction and transition. The design phase defines the scope of the project and develops a business case study. The development phase defines a plan of the project, specifying the characteristics and the underlying architecture. In the building phase the product is implemented and in the transition phase the product is transferred to users.

Taking into account the drawbacks of the waterfall model (mainly the lack of feedbacks from the users after the software development process was started) software engineering released a new development model, called spiral (see Figure 5), in which it is possible to move backward and forward between the various phases, as a consequence of user feedbacks.

This model encourages the incremental development (see Figure 6) in the software life cycle, and prototyping. In fact through the prototype it is possible to provide the user with an idea of how to run the application, and which functions are possible, so that it will be possible to cycle the development phases, introducing high level of flexibility and being able to change the initial requirements, as a result of the feedbacks received from the users.

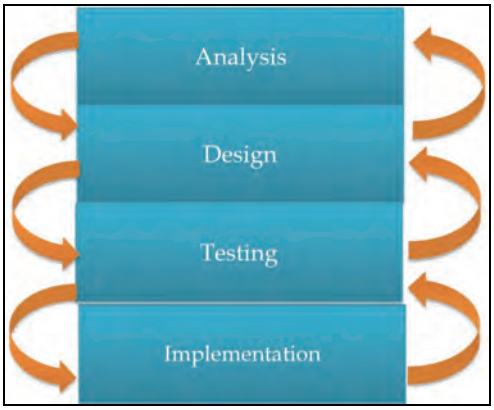

Fig. 5. Spiral Development Model.

These new paradigms adopted in the models related to the software development processes, as pointed out by Lorés & Granollers (op. cit.), also impose a change in the model of the user-application interaction, which forces the adoption of a new methodology in which the interaction with users is more natural, and more efficiently implemented, and which facilitates the comprehension of the system by new users and eliminates the inconsistencies in the interaction model.

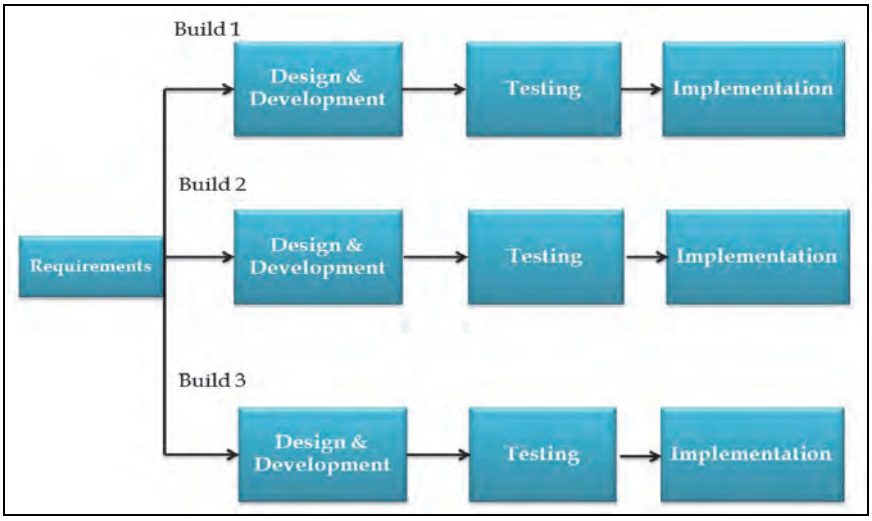

Fig. 6. Incremental Life Cycle Model.

4.3 The usability and the user centred design

The usability of software applications is an important branch of software engineering called usability engineering, which spans the "Human Machine Interaction" that helps the designers and the developers to release applications which are easy to use and understand, whose tasks may be carried out easily by the users, according to the principles introduced by cognitive psychologists (Norman, 1998).

Nielsen (1993), who was considered the "father" of usability, defines it as *a support to user tasks* (i.e.: it makes easier for people to do what they want to do). Merkovich (1999) expressed the needs of usability as a measure of its usefulness, its simplicity, its ease of learning and an assessment of a task, and a given context. Floría (2000) defines usability as the measure of how a product can be used by the users to achieve specific goals with effectiveness, efficiency and satisfaction in the context of a given use. The three authors agreed that usability has the purpose of facilitating the tasks of a user in a given context, which represents the user and the system in the environment in which she/he operates.

Usability, according to Floría (op. cit.), refers to the speed and ease with which people carry out their tasks through the use of the product, and involves the following concepts: (a) *an approach focused on the user:* to develop a usable product, designers and developers have to interact with people who represent the current or potential users of the product. (b) *develop a comprehensive knowledge of the context of use:* people use products to increase their own productivity. A product is considered easy to learn and use when the time required to the user to carry out her/his tasks is short, when the number of required steps is small, and when the probability of successfully carrying out the appropriate action is high; to develop usable products one has to understand the objectives of the user, and the type of work and tasks the product automates, modifies or embellishes. (c) the product has to meet the needs of the user: users are busy people trying to carry out a task. They will relate usability with productivity and quality. The hardware and software are tools that help busy people to carry out their tasks and to enjoy their leisure.

The International Organization for Standardization (ISO) provided two definitions of usability: the ISO/IEC 9241-11 (1998), defined the concept of usability as "*the extent to which a product can be used by certain users to achieve specific goals with effectiveness, efficiency and satisfaction in a context of specified use*". This definition is focused on the concept of usage quality, i.e.: refers to how effectively the user carries out specific tasks in specific scenarios. The ISO/IEC 9126-1 (2001) provided the following definition : "*usability refers to the ability of a software being understood, learnt, used and being attractive to the user, under specific conditions of use*". Manchón (2003) noted that this definition emphasizes the internal and external attributes of the product, which contribute to its usability, functionality and efficiency. He also noted that the usability depends not only on the product but also on the fundamental actor, the "user". The usability cannot be evaluated on an isolated product (Bevan (1994) quoted by Manchón, op. cit.).

The usability of a software application must be one of the driving criteria of the software quality assessment, and one of the main characteristics determining the user satisfaction using a new product.

Considering these issues related to the software development, the main target of a software developer is to implement software products very intuitively, inducing in the final user only

a small cognitive load to perform the required tasks. In fact, if an agile approach is adopted and if the user is involved from the beginning of the project, the various tasks the user has to carry out will be considered in all phases of the software development, and consequently the agility will become the integrating process for the user involvement and her/his satisfaction will be guaranteed independently from the specific technique, method or tool adopted. The requirement is to develop an agile and usable product.

4.3.1 Characteristics of usability

Nielsen (op. cit.) argued that usability is not a simple one-dimensional property of a user interface. Usability has multiple components and is associated with five (5) basic characteristics: learning ability, efficiency of use, capacity for memorization, bug tracking and user satisfaction. In some ways the learning capacity is the main attribute of usability. A system should be easy to learn so that the user can start any task using the system quickly.

It is also important that the various tasks have to be organized and documented so that they should be easy to remember, in this way even an occasional user can be productive relatively quickly. The system should also have a low error rate, so that the users can solve errors easily. Above all, it is important to prevent the system from catastrophic errors. And finally, the satisfaction of the user has to be taken into account.

The above mentioned features imply indirectly a reduction and a general optimization of production costs, as well as an increase in productivity. Usability allows the users to perform tasks faster. On the other hand, Merkovich (op. cit.) explained how the usability concept involves the usefulness, the ease of use, the ease of learning and the user's appreciation of the product.

The utility is the ability of a tool to help meet specific tasks and it is important to note that a tool which is very usable for one task, can be less usable for another, even though it is a task which is similar but not identical. The ease of use is related to the efficiency, measured as the rate of possible errors. A very easy to use tool will allow the user to carry out more operations per unit of time (or shortest time for the same operation) and will decrease the likelihood of errors occurring.

The ease of learning is a measure of the time required to carry out a task with a given degree of efficiency, achieving a degree of knowledge, even if the system will not be used for a while. While the ease of learning is usually directly related to the usability, this is not necessarily true. The ease of learning is a relative measure, as there are very complex systems that cannot be learned quickly.

Commercial software producers are implementing their own techniques for the design and the implementation of software. In some cases the usability principles are included, and even considered unavoidable. Floría (op. cit.) summarizes the main benefits originated from the adoption of the usability techniques in the design and implementation of software systems: a) *a reduction of the production costs*: such costs, in fact, can be reduced avoiding the over-design and the modifications required by the customers after the product has been released; b) *a reduction of the maintenance and support costs*: the usable systems require less training, a reduced support and maintenance actions; c) *a reduction of the usage costs*: the

systems fitting the user needs increase the productivity and the quality of the actions and of the decisions; the systems hard to use diminish the wellness and the motivation of the user, and may contribute to the increase of absenteeism from work. Furthermore the users waste more time using the system and they are not pushed to explore the advanced facilities of the system, which may not be used in the routinary actions; d) *improve the quality of the product*: the user-centred design turns out products with high quality and more easy to use, more competitive in a scenario were simple products are preferred.

4.3.2 Products usable on the basis of the principles of user-centred design

After having stressed the importance of the usability concept, particularly designing products, our attention is focused now on the importance of the user-centred design. Floría (op. cit.) stated that to achieve a high level of usability it is necessary to adapt the design process to the user-centred design principles, which represent a reformulation of the principles of classical ergonomics from which the accessibility guidelines are derived. The author presents the principles of user-centred design: 1) the control of the situation must be handled by the user; 2) it must be a direct approach; 3) the consistency is essential in the design part; 4) we have to enable the solution of errors; 5) we have to catch appropriate feedbacks from the users; 6) the aesthetics cannot be neglected; 7) the design should be characterized by simplicity; 8) it is essential to follow a rigorous design methodology; 9) the design team must be balanced in term of competencies; 10) there are four parts in the design process (analysis, design, implementation and test); 11) the usability concepts have to be taken into account during the design process; 12) the design has to be understood by users; 13) if the user pool is not enough satisfied by the design, the process has to be restarted from the beginning.

It should be noted that some of the principles of user-centred design are related to the principles of the heuristic evaluation, and both are directed to foster the usability of the systems. However, they clearly differ, since the user-centred design principles are used, despite the redundancy, in the design phase of a system, while the heuristic evaluation is used to measure the system's degree of usability, and to verify that it complies with the principles of user-centred design. The adoption of usability principles as driving concepts of a methodology for designing agile and usable software, may sort out a powerful tool for implementing an high quality product focused on user needs.

4.4 Standard ISO 9000-3: quality

Nowadays the major concern in software development is related to the quality of products. The users are expecting software products which have to be easy to learn and use, able to solve their needs and face their problems. Unfortunately, the software industry is often affected by huge problems like the high costs related to the refinement and the optimization of the released code, the time wasted correcting bugs, and the presumption of knowing all users needs. The agile software development process combined with the usability principles allow to prevent such problems by adopting good design and implementation strategies, and helping the designers and the code developers to fulfil the schedule and the budget constraints.

The software industry is adopting models to improve the quality of their operations and correct their failures. It is desirable they will implement a statistical analysis of the software

development process to monitor the activities, ensuring they are able to produce the same results. The adoption of an agile and usable approach and the quality control techniques will guarantee the successful planning of future projects, the costs optimization, the increase of the efficiency and the productivity, allowing to develop best quality products and to generate more benefits for the company.

The most popular quality control standard, the ISO 9000-3 released by the International Standards Organization (ISO), defines the guidelines for quality control, governing the implementation of the standard ISO 9001 to the development, provision and maintenance of software. It provides to both developers and customers, a set of guidelines for assessing the quality of software development processes.

The adoption by software companies of the standard ISO 9000-3 allows to: a) increase the competencies to afford the European market; (b) enable them to meet the client expectations; (c) obtain quality benefits and competitive advantages in the market; (d) adopt a clear market strategy; (e) reduce production costs. The benefits of obtaining the ISO 9000-3 certification, include the: a) increasing quality of the documentation systems; (b) positive cultural change induced in the employees; (c) increased efficiency and productivity; (d) increased perception of quality; (e) increased client satisfaction; (f) reduced customer quality audit; (g) time reduction of the system development.

The standard ISO 9000-3 is based on the assumption that following a well-defined software engineering strategy the company will be able to release higher quality software products, meeting the deadlines. The Software Engineering provides development models, methods and techniques to specify requirements, to establish the development plan, and to design, code, test, and document software products. In other words, it provides the instruments to release a software product according to the standard ISO 9000-3.

4.5 The usability engineering

The complexity of IT applications has stimulated the research on usability engineering, which in particular profits of the achievements of the human-machine interaction discipline. The usability engineering is a multidisciplinary field, combining expertise on computer science, psychology, linguistics, sociology, anthropology and industrial design. This term has been used since the mid of 1980s, to identify a new discipline which provides "*systematic methods and tools for the complex task of designing user interfaces easily understandable, quickly learnable and reliably operable*" (Buttle, 1996).

The importance of the user interface of a software application is known from the origin of the computer science, since the user is experiencing through it the benefits and the services of a software application. Most of the technical quality of the application appears through the user interface. If it is not effective, the functionality of the application and its usefulness are limited: the users are confused, frustrated and angry; the developers lose credibility and the company has to bear high costs and low productivity. The usability engineering aims to minimize the cognitive and perceptual overload of the user using an application. It uses a method of iterative design with rapid prototyping (support tools are essential), whose skeleton is represented by the cycle "analysis - design – implementation - evaluation" (see Figure 7), which is repeated several time, increasing progressively the system quality and its functions. The stage of evaluating the prototype, comparing its functions with the user expectations and needs, is a crucial phase for the success of the proposed approach.

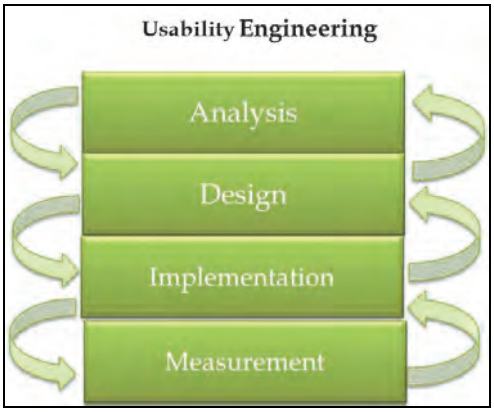

Fig. 7. Usability Engineering method.

In a proper usability engineering process the following steps should be carried out: a) define the goals of usability; (b) establish the planned usability levels to be achieved; (c) analyse the impact of different design solutions; (d) take into account the feedbacks from the users to design the product; (e) iterate through the cycle design-evaluation-redesign to achieve the planned user consensus and quality. Finally, in Figure 8 we summarize the evolution of the three methods and their interrelations respect to the traditional software development cycle, trying to include for the first time the usability engineering in the software development process.

5. Usability and the new engineering paradigm: agile usability

The usability is an attribute of software systems assessing their quality, which can be expressed as a sum of the ease of learning, the efficiency, the error recovery, and the satisfaction of the user (Nielsen, 1994). On the other hand, the user-centred design is a highly structured process, which is focused on the interpretation of needs and objectives of the user of a product. Therefore, if we consider the end user, the goal of the interaction design is to provide her/him useful features. This is why, when designing applications usable agile, we should consider that the interaction designers are focusing their attention on what is desirable for the user in terms of user interface functions, the interface developers are interested on what they are able to build for the application, and stakeholders (companies or users) on what is feasible starting from the experience of the user. The user experience in agile usable software development is important since: a) the end users make emphasis on the necessary functions which enable them to achieve their goals; (b) they facilitate the interface design, identifying its behaviour; (c) the experience of the users allow to define the formalism, using of the agile method. Such aspects, that are crucial for the group of designers and developers, are determining the success of the final product having included the users and their experiences in the application development cycle.

Agile usability is a recent concept developed by Nielsen & Norman (s. d.), who stated that the *agile* term is a new paradigm of software production. The *agile* term identifies a software development process which replaces in the software production the above mentioned classical cascade model (waterfall), as pointed out by Grezzi (2004). The agile model divides

into several cycles the typical cycle of a software development process. Therefore, the phases related to the analysis, the design, the implementation, the testing and the release of the software product are applied to a small group of functions (phase) and repeated for each of the subsequent phases. Each of these small and incremental phases, is called in agile terminology "sprint", as for the SCRUM method, by which we will assume the same term to describe the incremental aspects of the development team in each stage of the software development process. If the result of each particular sprint is not considered functionally "complete", it has to be included in the following sprints in order to fulfil the user expectations. Once a sprint has been completed, the development team has to reconsider the priorities of the project.

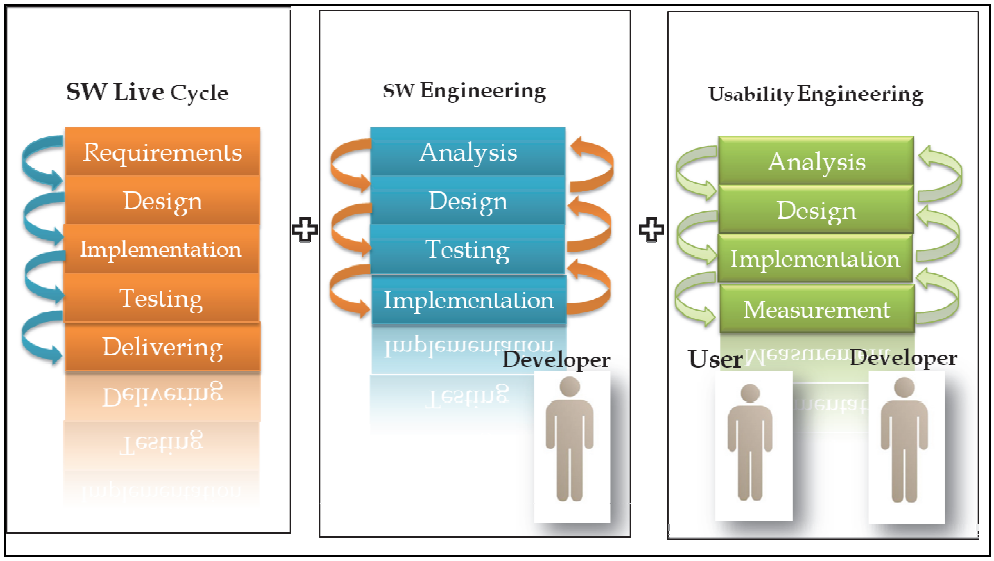

Fig. 8. The integration of the three models (Software life cycle, Software engineering and Usability engineering) to reach an optimal design and implementation of a software product.

Agile methods privileged the communication in real time, preferably face to face, respect to the written documentation. As described by Canós, Letelier & Penadés (s. d.), the agile development team is composed by those persons who are necessary for carrying out the software project. This team should at least include the designers and their customers. The aim is the customer satisfaction, not only the fulfilment of a contract. The use of agile methodologies, and in particular of the usable-agile method, has the purpose of minimizing the costs of software development. The most important principles of the agile methodology are expressed in the *Agile Manifesto* (Beck, Beedle, Cockburn et al., 2001): a) individuals and their interactions are more important than processes and tools: the relationships and communications between the actors of a software project are the principal resources of the project; (b) it is more important to have a working software respect to an excellent documentation: new versions of the software have to be released at frequent intervals, maintaining a simple and technically advanced code, minimizing the documentation; (c) it is

more important the collaboration with customers than the contract negotiation; (d) Responding to change is more important than following a plan: the development group has to be ready to find new solutions as soon as a change in the project occurs.

The term agile indicates all those development methodologies that transformed the old paradigms of Software Engineering (waterfall model, spiral model, etc.) based on a set of specifications and in a sequential structure of software development. This is common to a set of methodologies we can group together, such as eXtreme Programming (XP), SCRUM, Feature Driven Development, DSDM, Crystal, Lean Software Development, Agile Usability Engineering (AUE). Such methodologies are called agile because they allow to review and modify the set of specifications during the development phase, activating a strong exchange of information between the designers, the developers, and the users.

A group of developers who is using an agile approach must understand that the user-centred design and usability are explicit development methodologies and should therefore be included in the software production process.

Agile usability is a system for the evaluation of the usability of a software product to be carried out during the agile development of any application (also Web sites), performing usability tests as soon as each development phase has been completed (Nielsen, op. cit.). This makes a breakthrough in the quality of software, since the usability of a product is tested from the beginning of the development cycle, reducing the risk of releasing a product the users will not appreciate and use.

6. Methodology AGILUSAB

We present a methodology for developing usable and agile applications, which takes into account the quality of software standards, the usability engineering and considers the interaction of the development team with the users a fundamental phase for a successful development of any software application. AGILUSAB is an incremental and iterative methodology proposed to develop software, with a particular attention to web applications.

6.1 Steps of the method

AGILUSAB, shown in Figure 9, is the methodology presented as a series of steps of comprehensive development phases, starting from the analysis, the design, the development of the user interface, the testing and the release of the application. AGILUSAB is an agile methodology for the development of interfaces and for the usability evaluation of the final products, in which the users and the customers play a crucial role being included in all phases of the development cycle.

AGILUSAB is an iterative Software Engineering method (Grezzi, 2004), it works mainly on simple interfaces (like a Web page) and provides for each development cycle the usability evaluation of each module, allowing the redesign of the proposed interface if considered necessary as a consequence of the assessment made.

The component of the method related to the usability is divided into three well known phases, which can be precisely evaluated: inspection, inquiry and testing. During the inspection the interface is analysed using empirical methods, like the heuristics of Nielsen, the cognitive paths (cognitive walkthrough), or other techniques the team of designers,

developers and usability experts are considering convenient to use. The inquiry phase uses techniques to assess the usability of the operations the user or the customer may carry out using the interface; these techniques might include: questionnaires, observation, focus group, thinking aloud, among others. In the testing phase potential representative users are considered to analyse how they carry out their tasks using the system (or a prototype of it), and the comments of the usability evaluators are taken into account; among the most commonly used techniques are: measures of performance, protocols of expression of the user, remote testing, retrospective testing, and alternatives to the classic usability testing, such as: coaching method, shadowing method, teaching method, Co-discovery method, retrospective testing. Finally, the results obtained in the various phases of the usability evaluation process are discussed with the panel of evaluators to obtain an overview of the assessment made. The part of the method that refers to the development of a given implementation, forms a cycle consisting of six phases: analysis, design, prototyping, implementation, testing and release.

Analysis: The agile development methodology starts with the analysis phase, whose objective is to show the behaviour of the interface through the use of case diagrams (see Figure 10), a very popular technique of Software Engineering, introduced by the Unified Modelling Language (UML). Once having shown the behaviour of the interface to the users, a dialogue with them is established. On the basis of the decisions taken by the developer team, a (low, medium or high-fidelity) prototype in paper or software is shown to the user, showing the behaviour of the new component. It is initially recommended to show a low-fidelity prototype. If no modifications are necessary, the user is informed that on the basis of her/his observations this phase can be closed, otherwise the phase is repeated from the beginning. The implemented prototype is then passed to the design phase.

Design: in the analysis phase the behaviour of the interface is analysed, while in the design phase the coherence of the various possible operations is carried out. At the beginning of this phase the logical structure of the interface has to be defined, sketching the various actions a potential user can carry out when interacting with it. To represent these actions, we use the sequence diagram technique provided by UML. To have an idea of the logical structure of the interface, the logical representation in term of the present classes (tables, objects…) is added to the sequence diagram. To explore all possible connections the navigation tree will be used, which consists on a highly connected graph having as root the considered interface and as sons the various reachable interfaces, as shown in Figure 11. Once the analysis of the behaviour of the interface have been carried out, the team will meet again with the pool of users and customers in order to asses the obtained results, evaluating the opportunity of redesigning the interface. In this case a new prototype will be defined, according to the users comments and requirements.

Prototyping: The prototype has to be designed during the analysis and design phases, as described in the previous two paragraphs. Once the prototype behaviour has been defined, it has to be implemented according to the principle of the "Agile Manifesto". Once the implemented prototype is approved by the users, the customers and the development group, the implementation phase can be started.

Implementation: this phase runs as the implementation phase of the classical Software Engineering approach, since it is not necessary the dialogue with the customers and the users (they are not supposed to be expert programmers and therefore their contribution is useless).

Testing: during this phase, the team of developers has to select a community of potential typical users and test the functionality of the interface. The test is useful to assess the improvements made by the different usability evaluation methods adopted in each stage of the development. A good approach could be to select users involved in some of the usability tests carried out in the previous phases, to observe her/his reaction acting on an already operational interface. Once an interface has been tested, there are three possible actions: a) if the test was not satisfactory, the cycle is repeated taking into account the users observations and fixing the errors; (b) if the test was successful and if a new interface has to be implemented, a new cycle will be started; (c) If all components have been implemented and tested, the release phase will be started.

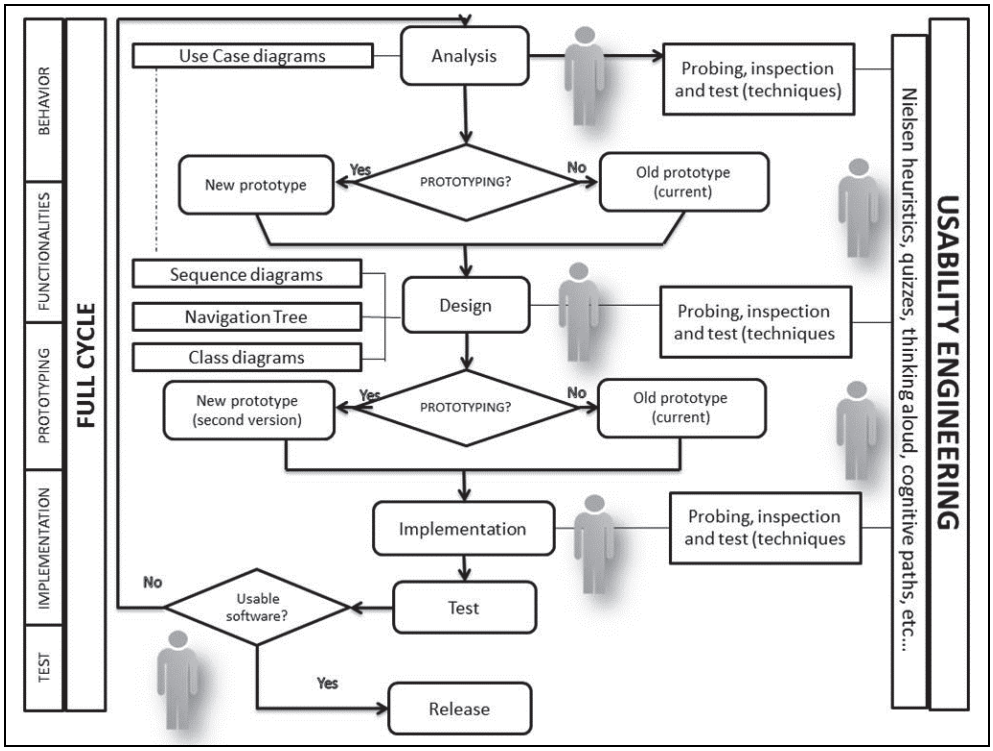

Fig. 9. Method AGILUSAB.

Release: the last phase of AGILUSAB is done only once, after all necessary cycles and iterations have been completed successfully. With the release phase the software is finally validated by the development team, the users and the customers, and is released for production.

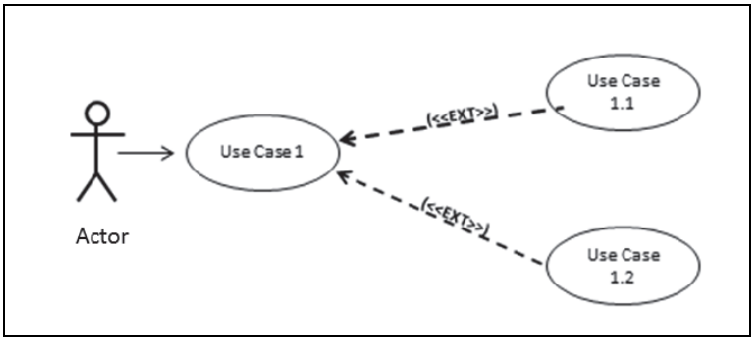

Fig. 10. Sample use case with an actor.

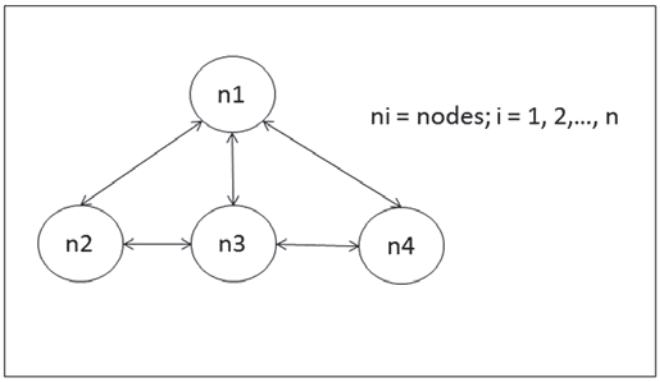

Fig. 11. Navigation tree.

7. Results: AGILUSAB used to restructure a web application

In order to test the AGILUSAB approach we evaluated the usability of the software product related to the web site of the travel agency MAVITUR (http://www.maviturviaggi.com), analyzing the XHTML pages and evaluating the usability of the software product. We applied our techniques and models (mainly the use case diagram and the sequence diagram) to identify, if applicable, the components with lack of usability and applying in such case the agile usability approach to restore its expected behaviour and functionality.

At the end of the evaluation process the team (composed by usability experts, final users and customers) expressed the evaluation of each interface. We are summarizing the results considering for each cycle only one interface of the software application.

HOME PAGE (index.html)

The inspection and the evaluation results are summarized in Table 1. The priority of the action required are expressed in the following way: 1= high, 2 = medium, 3 = low.

In Figure 12 the use case diagram of the home page is presented, related to the *analysis* phase. In Figure 13 the sequence diagram is presented and in Figure 14 the navigation tree

of the same page is shown; both figures are related to the *design* phase. In conclusion, the team, considered the high number of usability problems detected in the current version, decided to design a new prototype according to the agile usability approach. In particular, new functions were planned in order to facilitate the navigation of the web site.

7.1 Web site evaluation using the closed questionnaire technique

We defined a questionnaire containing 32 general questions related to the web site. The responses were collected and evaluated using the open source package LimeSurvey, a very efficient tool for facilitating the collection of a suitable statistical sample and able to perform the descriptive statistics over the collected responses. The statistical sample were made by 95 users of the site, selected among customers and potential users. In Figure 15 an excerpt of the questionnaire is shown and in Figure 16 the summary of responses to a given question processed by LimeSurvey is shown.

The questionnaire responses made by the users were highly considered by the team, since they allow to catch wishes and expectations of the potential users of the web site. If a given problem were experienced by the 70% or more of the users, the assigned priority were 1, while if only few of them noticed a problem, this was rated with priority 3 and in the remaining cases the assigned priority were 2.

FAMILY (CATEGORY)	PROBLEM	COMMENT	PRIORITY
Design (graphic)	*Click* unknown	The user does not quickly understood that the photo gallery is located at the top right and she/he can click on it.	**1**
Design (layout)	*Frozen layout*	Small schemes that require horizontal scrolling	**2**
Design (scrolling)	*Scrolling*	The displacement was hiding significant areas of the page	**1**
availability *(navigation)*	*Advanced technologies and Plugin*	The lack of Flash Player damaged the graphic aspect of the site and prevented the access to the gallery.	**2**
Information (content)	*bums contents*	Missing text explaining the nature of the site and the agency.	**1**
Support (flow problems)	*User guide*	The sections "News" and "Events" are not important.	**1**
Design (graphic)	*Violation of the Web rules*	The hyperlink structure do not follow the Web rules	**3**
Information (content)	*Arrangement of contents*	The navigation menu is not arranged properly: it highlighted the external appearance of the agency and not what the user really is looking for.	**1**
Information (content)	*Fulfilment of the user needs*	The organization of the content is not suitable for fulfilling the user expectations visiting a travel agency website.	**1**
Design (typography)	*Font colour and background*	The colours are not appropriate and do not motivate the user to visit the offered services.	**1**

Table 1. Evaluation results related to the home page of the inspected site.

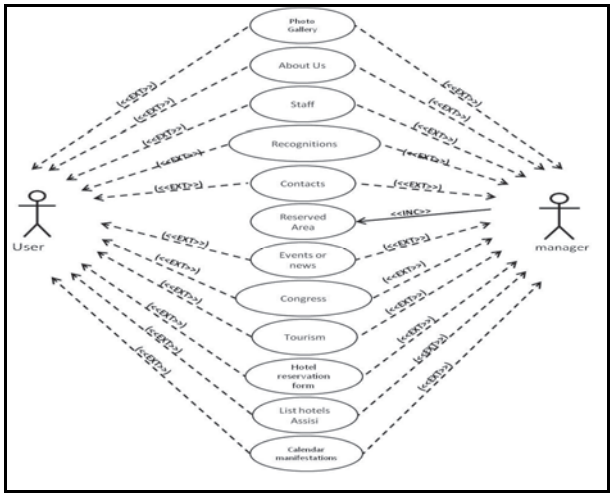

Fig. 12. Use case of the home page of Mavitur travel agency.

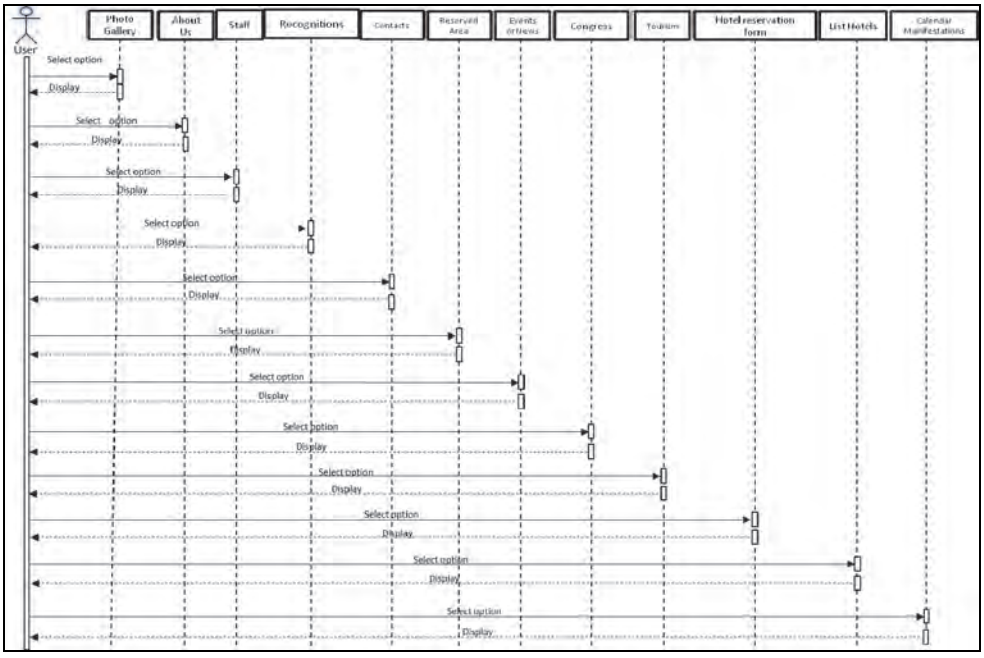

Fig. 13. Sequence diagram of the home page of Mavitur travel agency.

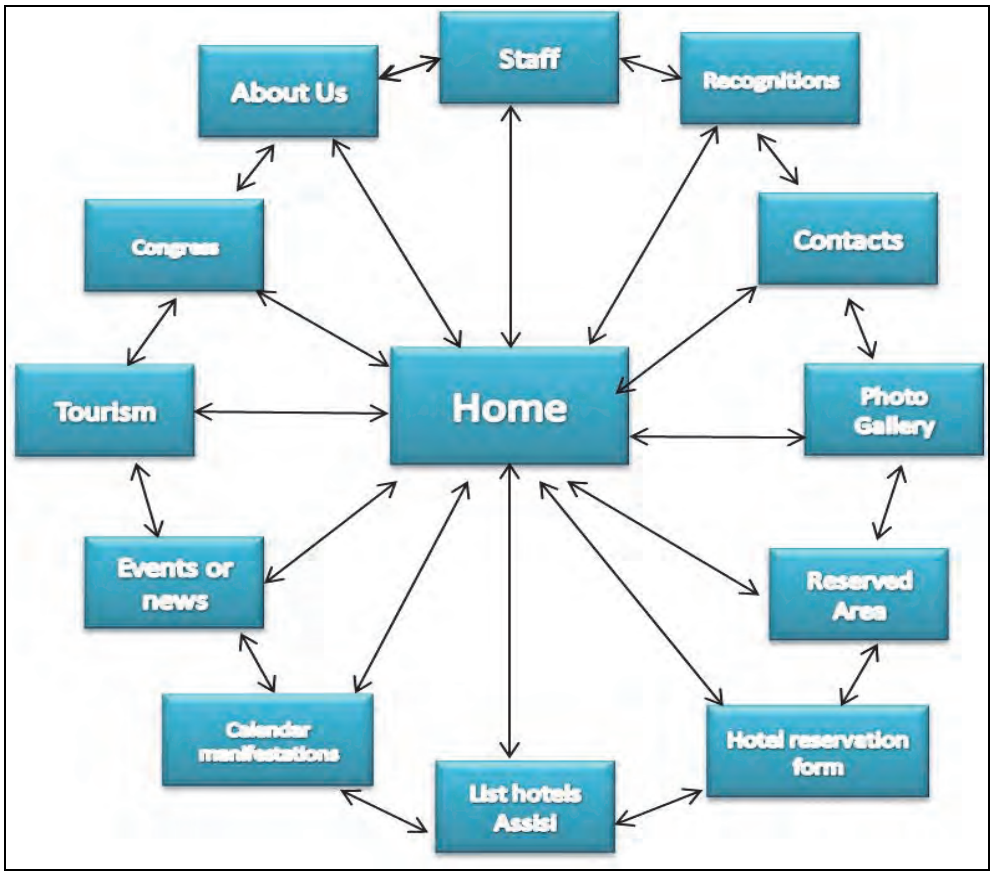

Fig. 14. Navigation tree of the interface version 1.0 of the home page of Mavitur travel agency.

Some responses provided by users were useful to identify a few usability problems not detected previously by the team. In Table 2 the detected problems are shown. The assigned priority was proportional to the frequency of the response among users.

The final decision of the team, at the end of the usability tests of the web site, were to redesign the site from scratch, considering the serious usability problems detected. In the following paragraphs the various milestones of the project will be analysed considering the implementation issues and the tests made with the users according to the agile methodology.

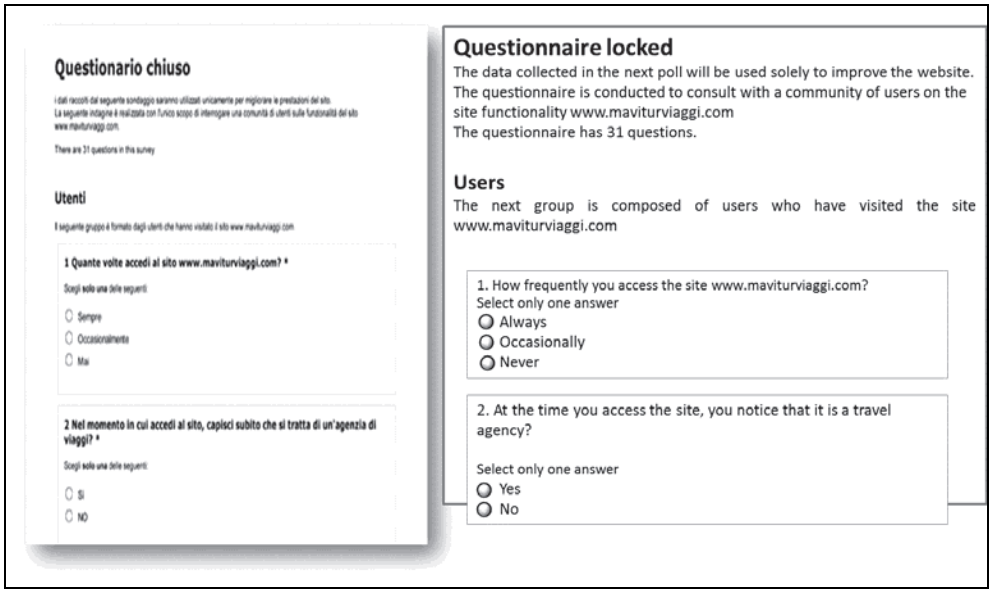

Fig. 15. Questions in the questionnaire (Original in Italian, English translation).

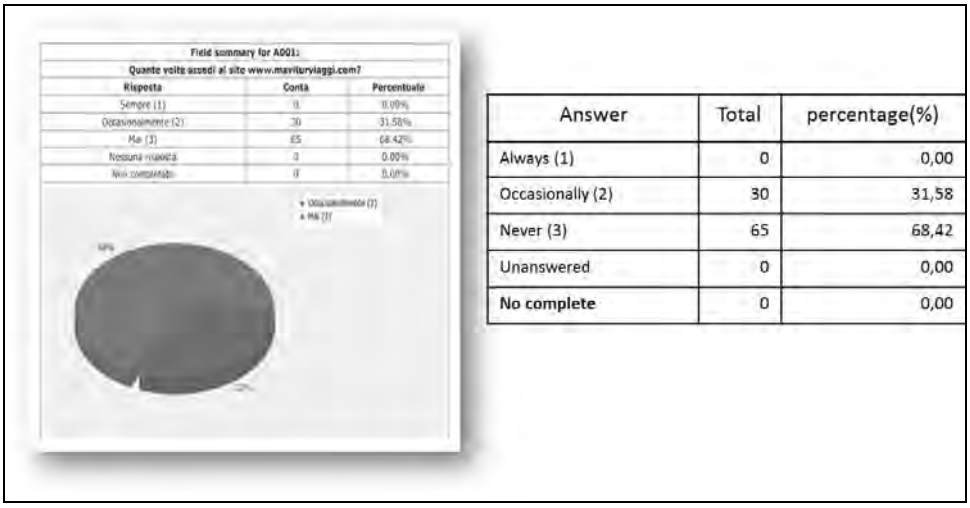

Fig. 16. Results of the questionnaire obtained through the application LimeSurvey.

In order to present the results of the research made by our team we will describe the behaviour of the web site, its functions and the resulting graphical interface. In particular our attention will be focused on the home page of the web site. This page has been completely redesigned, optimizing its behaviour and increasing its functions, as a consequence of the performed usability tests. The team introduced a hierarchical menu, structured with submenus in order to improve the usability of the web site.

FAMILY (CATEGORY)	PROBLEM	COMMENT	PRIORITY
Design (graphic)	*Click* unknown	The buttons on the home page does not invite the user to click; this fact does not help the user to comprehend the functionality of the site.	**2**
Availability (Information Architecture (IA))	*Specificity of the link, labels and buttons*	The buttons on the site do not represent the assigned function.	**3**
Information (content)	*Satisfying customer needs*	The map does not show the location of the agency nor the hotels of Assisi.	**1**
Design (Writing)	*Poor wording*	The texts on the site will not attract customers to usufruct of the services offered by the agency.	**3**
Support (flow problems)	Guide users	The arrangement of the contents can confuse users and prevent run some actions.	**3**
Search	*Irrelevant the search site*	Missing search function on the site.	**3**
Information (content)	*Satisfying customer needs*	The site design is not consonant with what users expect from a travel agency.	**1**
Information (content)	*Satisfying customer needs*	Reservation cards do not have a validation application to filter bad data.	**1**

Table 2. Report of the concerns expressed by the users after having accessed the original web site.

In Figure 17 the use case resulting from the new design is shown. Figure 18 is describing the functions of the web site through the sequence diagram. In Figure 19 the resulting navigation tree is shown. The class diagram is not applicable in the present case, since a database is not necessary.

In Figure 20 the graphical interface resulting from the application of the usability guidelines and taking into account the suggestions made by the users after having accessed the original web site, is shown. In particular all colours have been changed in order to match the colours of the company logo. The contrast between the text and the background has been optimized taking into account users with visibility problems. We maintained the graphic design of the original version of the web site.

After having defined the prototype of the home page we afforded the implementation and testing phases. Since the implementation cannot be described here, we detail the testing phase, to prove that the usability guidelines allowed to implement a convenient web site, more usable and accessible.

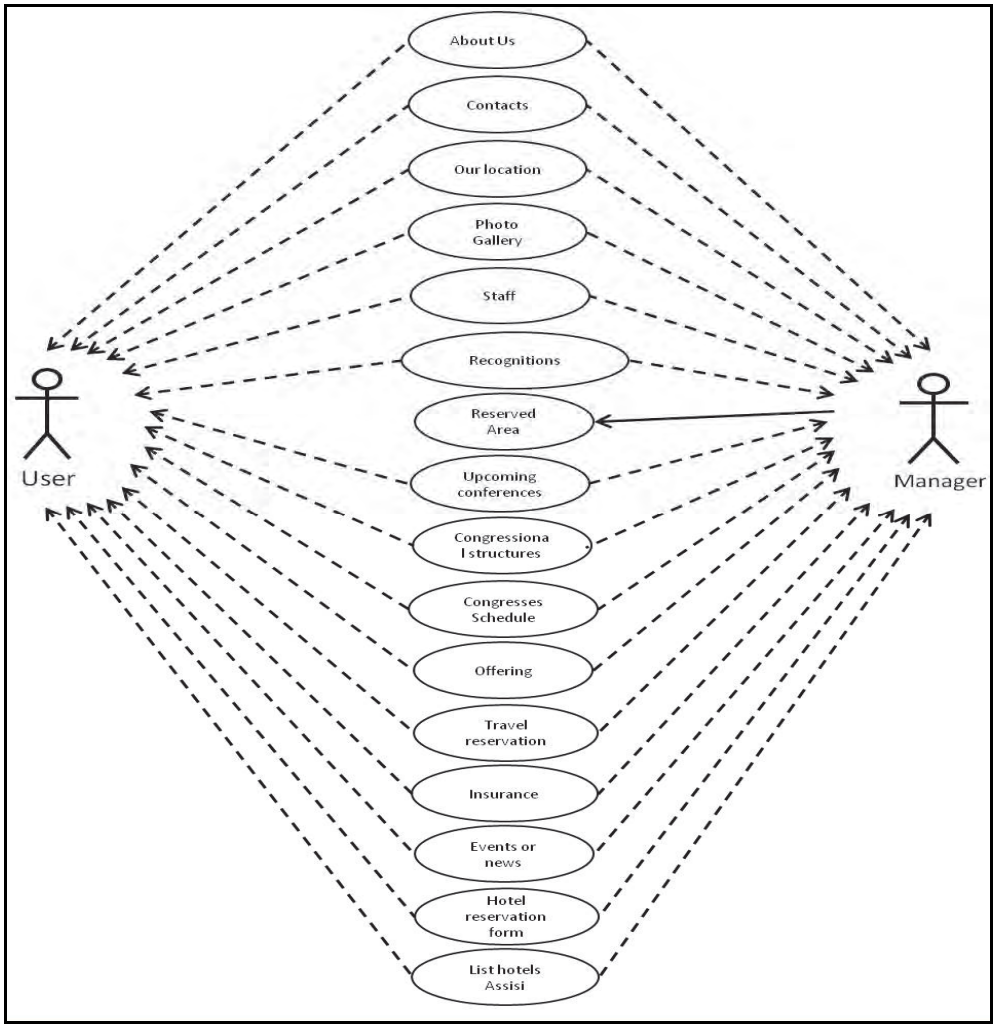

Fig. 17. Use Case diagram of section Home Page v 2.0.

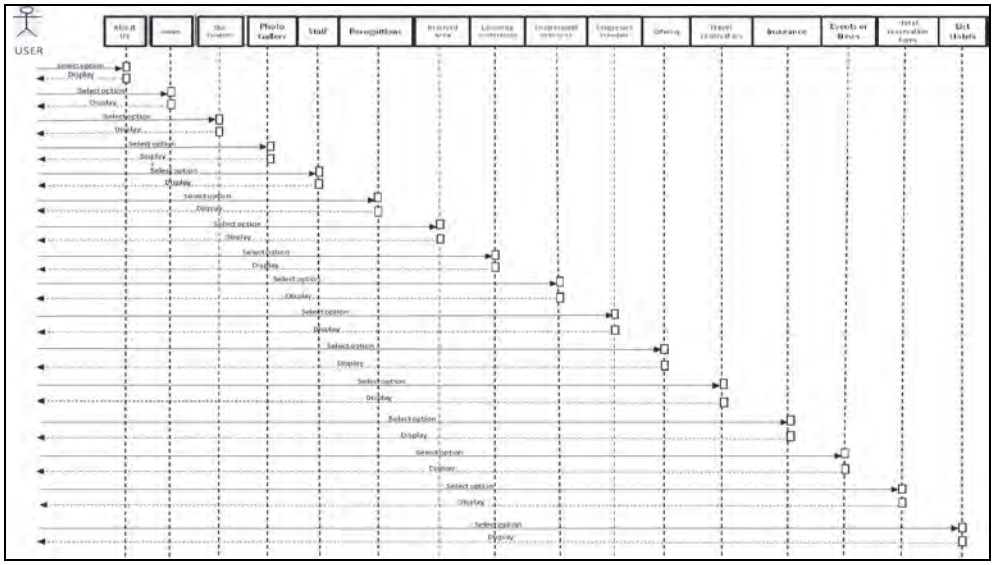

Fig. 18. Sequence Diagram Home Page v2.0.

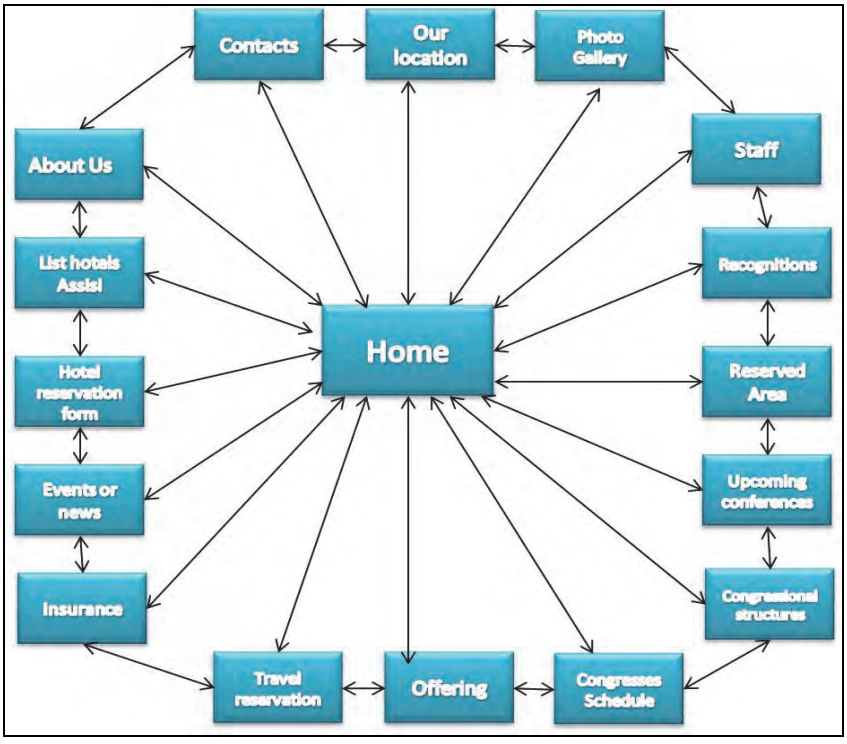

Fig. 19. Navigation Tree Home Page v 2.0.

To this purpose we repeated the usability test on the new graphical interface. The new test were run on a set of 10 users, 5 of them were aware of the features of the original web site. The results were really good, since all users appreciated the usability and the ease of use of the new graphical interface, proving that our method, based on the agile usability approach, is efficient and helps the developers to implement usable and simple software products.

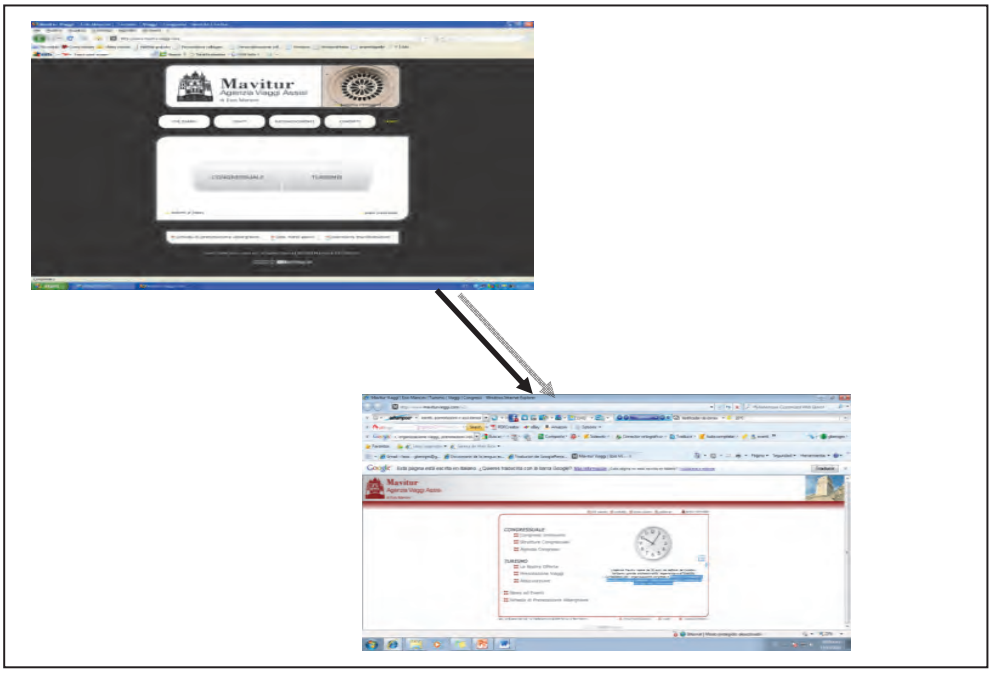

Fig. 20. GUI Home Page v 2.0.

8. Conclusions

At the present time the Human Machine Interaction has taken a central role in computer science, since it is focused on the importance of users and their experiences in all phases of software development.

We presented our AGILUSAB method, based on the agile and the usable methodologies and we presented the results obtained applying our method to a web site of a travel agency located in Assisi (PG), Italy. The AGILUSAB method allowed to redesign the web site solving all issues related to the lack of usability and the poor access to the relevant information. The coherence introduced in the navigation tree, the harmony of colours, make the user interaction easier and more effective.

Software products released according to the AGILUSAB method will minimize the cognitive load of the user, will minimize the errors and stimulate the user to interact with the site, successfully profiting of the information available. The collaboration between the stakeholders (end-users, customers, project managers, designers, architects, analysts, developers) allows to release products highly competitive and adaptable to their needs.

We want to add another important benefit related to the AGILUSAB method, which is related to social and economical issues. As per the social issues we would note that the constant evolution of the Internet, the inclusion of social networks and increasingly cheaper prices of computer products, enabled a large number of people to use the computer and access the global network. The diffusion of agile and usable products will enable people to use the technology in a more productive and efficient way. As per the economic issues, the diffusion of agile and usable methods will optimize the software life cycle, reducing the time required for releasing the final product and reducing the development costs, since the final users are deeply involved in all phases of the software production.

For computer science professionals it is crucial to approach the psychology of customers and users, increasing the capability of establishing a dialogue with them. The strong competition we are observing nowadays is claiming new approaches and innovative strategies to catch the user expectations and wishes. The usability of the released software represents an important criterion for a positive evaluation of a company and its employees.

9. References

Benigni, G., Gervasi, O., Ordaz, J., Pallottelli, S. *Usabilidad ágil y reingeniería de sitios web: Usableweb.* Saber. Volume 3 – 2011, in press, Venezuela.

Benigni, G., Gervasi, O., Passeri, L., Tai-Hoon, K. (2010). *USABAGILE Web: A Web Agile Usability Approach for Web Site Design*. Lecture Notes in Computer Science, Volume: 6017, Publisher: Springer Berlin Heidelberg, pages 422-431, March 2010, Germany.

Beck, K., Beedle, M., Cockburn, A. et. al. (2001). *Manifesto for agile software development.* Date of access [April, 2011]. Available from: <http://www.agilemanifesto.org/principles.html>.

Butler K.A. *Usability Engineering turns 10.* Interactions, June 1996.

Canós, J., Letelier, P. y Penadés, Mª. (s. f.). *Metodologías ágiles en el desarrollo de software*. Date of access [February, 2011]. Available from: http://www.willydev.net/descargas/prev/TodoAgil.pdf.

Floría, A. (2000). *Recopilación de Métodos de Usabilidad*. Date of access [March, 2011]. Available from: <http://www.sidar.org/visitable/Herramientas.htm>.

Grezzi, C. (2004) *"Ingegneria del software, fondamenti e principi"*, Editorial Pearson.

ISO/IEC 9241-11. (1998). *Guidance on usability.*

ISO/IEC 9126-1(2001). *Software engineering – Product quality -- Part 1: Quality model.*

Jakob Nielsen, (2003). *Usability Engineering*, Academic Press, USA.

Jacobson, I. (1998). *Object-Oriented Software Engineering*. USA: Addison Wesley.

Lorés, J., Granollers, T. (s. f.). *La Ingeniería de la Usabilidad y de la Accesibilidad aplicada al diseño y desarrollo de sitios web*. Date of access [February, 2011]. Available from: <http://es.scribd.com/usabilidad-y-diseno/d/19452307>

Manchón, E. (2003).*¿Qué es la usabilidad? Definición de Usabilidad.* Date of access [February, 2010]. Available from: <http://www.alzado.org/articulo.php?id_art=39>

Manning, L. (1992). *Introducción a la neuropsicología clásica y cognitiva del lenguaje*. Madrid: Trotta.

Martínez de la Teja, G. (2007). *Ergonomía e interfases de interacción humano-computadora*. IX Congreso Internacional de Ergonomía. Date of access [April, 2011]. Available from: http://www.semac.org.mx/archivos/9-6.pdf.

Merkovich, E. (1999). *La Intersección entre Factores Humanos, Diseño Gráfico, Interacción y Comunicación.* Date of access [March, 2011]. Available from: <http://fractal.gaiasur.com.ar/infoteca/siggraph99/diseno-de-interfaces-y-usabilidad.html>.

Nielsen, J. (1993). *Usability Engineering*. Academic Press, Inc. Boston.

Nielsen, J. (2008). *Agile Development Projects and Usability.* Date of access [February, 2011]. Available from: <http://www.useit.com/alertbox/agile-methods.html>

Nielsen, J. & Norman, D. (s. f.). *Agile Usability: Best Practices for User Experience on Agile Development Projects*. 2nd edition. Nielsen Norman Group Report.

Norman, D. A. (1988). *The psychology of everyday things,* ISBN-13 978-0-465-06710-7 Basic Books, New York, NY

Pressman, R. (2005). *Software Engineering: A Practitioner's Approach.* 5th edition. McGraw-Hill Higher Education.

Rodríguez, L. (n. d.). *Scrum, una metodología Ágil (I).* Date of access [February 19, 2011], Available from: http://es.debugmodeon.com/articulo/scrum-una-metodologia-agil-i.

Schwaber, K. & Beedle, K. (2002). Agile Software Development with Scrum. 1st Edition. ISBN: 0130676349, ISBN-13: 9780130676344. Prentice Hall.

Wittawat, Ch. (2010). *Software engineering, Software Document, Software Process.* Date of access [February, 2011]. Available from: *http://software-document.blogspot.com/2010/06/spiral-model.html*

Zavala, R. (2000). *Diseño de un Sistema de Información Geográfica sobre internet*. Date of access [Mach, 2011]. Available from: <http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html>.