Supporting Work Practice through
End User Development Environments

M. F. Costabile¹, D. Fogli², R. Lanzilotti¹, P. Mussio³, A. Piccinno¹

¹ Dipartimento di Informatica, Università di Bari, Bari, Italy
² Dipartimento di Elettronica per l’Automazione, Università di Brescia, Brescia, Italy
³ Dipartimento di Informatica e Comunicazione, Università di Milano, Milano, Italy

{costabile, lanzilotti, piccinno}@di.uniba.it, fogli@ing.unibs.it, mussio@dico.unimi.it

RAPPORTO TECNICO 01.10.2005 10 ottobre 2005

ABSTRACT

End User Development means the active participation of end users in the software development process. In this perspective, tasks that are traditionally performed by professional software developers are transferred to end users, who need to be specifically supported in performing these tasks. We have developed a methodology that supports user work practice and meta-design, allowing experts in a domain to personalize and evolve their own software environments. In this paper we illustrate how this methodology is applied to a project for the development of an interactive system in the medical domain. Physicians and their activities have been carefully analyzed through a field study that is reported in the paper, in order to provide them with computer systems that may improve their work practice and determine an increase in their productivity and performance, i.e. a better quality of diagnosis and medical cure, with the achievement of competitive advantage for the organization they work in.
Supporting Work Practice through
End User Development Environments

M. F. Costabile¹, D. Fogli², R. Lanzilotti¹, P. Mussio³, A. Piccinno¹
¹ Dipartimento di Informatica, Università di Bari, Bari, Italy
² Dipartimento di Elettronica per l’Automazione, Università di Brescia, Brescia, Italy
³ Dipartimento di Informatica e Comunicazione, Università di Milano, Milano, Italy

{costabile, lanzilotti, piccinno}@di.uniba.it, fogli@ing.unibs.it, mussio@dico.unimi.it

Abstract

End User Development means the active participation of end users in the software development process. In this perspective, tasks that are traditionally performed by professional software developers are transferred to end users, who need to be specifically supported in performing these tasks. We have developed a methodology that supports user work practice and meta-design, allowing experts in a domain to personalize and evolve their own software environments. In this paper we illustrate how this methodology is applied to a project for the development of an interactive system in the medical domain. Physicians and their activities have been carefully analyzed through a field study that is reported in the paper, in order to provide them with computer systems that may improve their work practice and determine an increase in their productivity and performance, i.e. a better quality of diagnosis and medical cure, with the achievement of competitive advantage for the organization they work in.

1. Introduction and motivation

The current strong technological push creates many expectations about the possibilities offered by software systems. Computer users are evolving from passive consumers of computer tools and data to a more active role of information and software artefacts producers (Fisher, 2000). This is also highlighted by Schneiderman’s claim that “The old computing is about what computers can do, the new computing is about what people can do” (Shneiderman, 2002). The “interaction” dimension in software systems pays much attention on the human side and forces to go beyond the traditional Turing machine approach to computation by considering interaction as a key feature (Wegner and Goldin, 2003).
The interaction dimension also creates new challenges for system specification, design and implementation. It is well known that “using the system changes the users, and as they change they will use the system in new ways” (Nielsen, 1993). These new uses of the system make the working environment and organization evolve, and force the designers to adapt the system to the evolved user, organization and environment. Current techniques for software specification and design, such as UML, are very useful for software engineers, but they are often alien to users’ experience, language, and background. Software development life cycles should foresee participatory (Schuler and Namioka, 1993) and open-ended design (Hix and Hartson, 1993).

In our experience as computer scientists, we cooperate in participatory design projects to develop computer systems to support professional people in their work practice. Such people, such as physicians, geologists, mechanical engineers, are experts in a specific discipline (e.g. medicine, geology, etc.), not necessarily experts in computer science. Moreover, their work practice consist in “unfolding activity in actual communities that is concrete and situated, complexly socially organised and technologically mediated” (Karasti, 2001).

Developing their tasks, professional people reason and communicate with each other through documents, expressed using notations, which represent abstract or concrete concepts, prescriptions, results of activities. Often, dialects arise in a community, because the notation is used in different practical situations and environments. For example, mechanical drawings are organized according to rules, which are different in Europe and in USA. These professionals need to use computer systems to perform their work tasks exploiting all the communication and operation possibilities offered by these systems, but they are not and do not want to become computer experts. Our work addresses this particular community of end users: in this paper, the word "end user" denotes experts in a specific discipline (e.g. medicine, geology), in general not experts in computer science nor willing to be, who use computer environments to perform their daily tasks. These end users often complain about the systems they use, and feel frustrated because of the difficulties they encounter interacting with them.

Our approach to system development starts from the observation of activities of such kind of expert users during their daily work. The research we have developed in this field, and the experience gained has brought us to develop software environments that supports users in performing activities in their specific domains, but also allow them to tailor these environments for better adapting to their needs, and even to create or modify software
artefacts. The latter are defined activities of End User Development (EUD), to which a lot of attention is currently devoted by various researchers in Europe and all over in the world (Sutcliffe and Mehandjiev, 2004; Burnett et al., 2004; Fischer and Giaccardi 2005; Myers et al., 2003).

EUD means the active participation of end users in the software development process. In this perspective, tasks that are traditionally performed by professional software developers are transferred to the users, who need to be specifically supported in performing these tasks. The active user participation in the software development process can range from providing information about requirements, use cases and tasks, including participatory design, to end user programming. Some EUD-oriented techniques have already been adopted by software for the mass market such as the adaptive menus in MS Word™ or some Programming by Example techniques in MS Excel™. However, we are still quite far from their systematic adoption.

EUD is a huge challenge in the current software engineering scenario. Somebody even thinks, for various reasons, that allowing end users to perform design activities is just an utopia. Ben Shneiderman, introducing the Lieberman book on Programming by Example (Lieberman, 2001), states that there is magic and power in creating programs by direct manipulation activities, as opposed to writing code. The eighteenth-century scientists, such as Ben Franklin, experimented with electricity and found its properties quite amazing. Successively, Franklin and other scientists, as Michael Faraday, laid the foundation for Thomas Edison’s diverse applications, such as generators and electric lighting. Thus, in Schneiderman’s opinion, also in the field of Programming by Example there are a lot of researchers like Franklin and Faraday who are laying the foundation for the Thomas Edison still to come, but it is difficult at the moment to tell which idea will trigger broad dissemination or which insights will spark a new industry.

On the other hand, some drawbacks can affect EUD: for example, if some situated practices of end users are inadequate, then they are replicated in the software environment obtained through EUD activities. However, end users are the owners of the problems, and the end user community is the referee of the adequacy of the work practice. Our approach develops tools to support end user work practice. These tools force end users to externalize the decision process and to document it through annotations. These annotations allow a critical evaluation of the work practice by the whole community. If the work practice of the
end user is inadequate within the professional domain, the inadequacies are made evident by
the annotations and can be discussed at large by the community.

In this paper, we illustrate our methodology for the design of software environments
describing its application to a project in the medical domain. The work described introduces
a perspective on system personalization, distinguishing between customization and tailoring
of software environments. The software environments are customized by the design team to
the work context, culture, experience, and skills of the user communities; they are also
tailorable by end users at runtime in order to adapt them to the specific work situation and
users’ preferences and habits. The aim is to provide the physicians with software
environments that are first of all easy to use and adequate to their tasks. The approach is
open-ended: we describe here the step currently being performed. In this step, physicians and
their activities have been carefully analyzed through a field study that is reported in the
paper, in order to provide them with computer systems that may determine an increase in
their productivity and performance, i.e. a better quality of diagnosis and medical cure, with
the achievement of competitive advantage for the organization they work in.

The paper is organized as follows. Section 2 presents the methodological assumptions
underlying our work. Section 3 recalls our (meta-)design methodology and Section 4
suggests some refinement with respect to the definition of meta-design provided in literature.
Section 5 describes the application of the methodology to a project in the medical domain.
Section 6 provides a comparison with related works and Section 7 concludes the paper.

2. Methodological assumptions

Following Schön (1983), we assume that end users do perform their activity as
competent practitioners, in that “they exhibit a kind of knowing in practice, most of which is
tacit” and they “reveal a capacity for reflection on their intuitive knowing in the midst of
action and sometimes use this capacity to cope with the unique, uncertain, and conflicted
situations of practice”.

Tacit knowledge consists of habits and culture that we do not recognize in ourselves,
but which can be used in performing our activities (Polanyi, 1967). End users, as competent
practitioners, apply their tacit knowledge if the current context and the tools at hand support
them to apply it.

Competent practitioners reason and communicate with each other through documents,
expressed using notations, which represent abstract or concrete concepts, prescriptions,
results of activities. Such notations emerge from users’ practical experiences in their specific
domain of activity. As suggested in (Petre, 1995), they highlight those kinds of information users consider important for achieving their tasks, even at the expense of obscuring other kinds, and facilitate the problem solving strategies, adopted in the specific user community. Notations reflect tacit knowledge shared among users in the community. Authors build documents in the notation using their tacit knowledge, and the document can be understood by readers, who possess a similar knowledge. The document conveys implicit information, which can be elicited only by readers in the community. Practitioners should be able to use their tacit knowledge also interacting with computer systems. The messages on which interaction is based should convey implicit information.

Users often complain about the system they use to perform their work tasks, but they are not and do not want to become computer experts. One of the reason that interactive systems are unusable is because the system imposes strategies of task execution alien to users, it drives them to follow unfamiliar reasoning strategies and to adopt inefficient procedures. It is well known that “using the system changes the users, and as they change they will use the system in new ways” (Nielsen, 1993). In turn, the designer must evolve the system to adapt it to its new usages; this phenomenon is called co-evolution of users and systems (Arondi et al., 2002). In (Bourguin et al., 2001), it is observed that these new uses of the system determine the evolution of the user culture and of her/his models and procedures of task evolution, while the requests from users force the evolution of the whole technology supporting interaction.

In order to design a system that meets users’ needs and expectations, we must take into account the following observations:

1. The notations developed by the user communities from their work practice are not defined according to computer science formalisms but they are concrete and situated in the specific context, in that they are based on icons, symbols and words that resemble and schematise the tools and the entities which are to be operated in the working environment. Such notations emerge from users’ practical experiences in their specific domain of activity. They highlight those kinds of information users consider important for achieving their tasks, even at the expense of obscuring other kinds, and facilitate the problem solving strategies, adopted in the specific user community.

2. Software systems are in general designed without taking into account the problem of implicit information, user articulatory skills and tacit knowledge, so that they can be interpreted with high cognitive costs. Actually, these factors are among those ones determining the so-called invisible work (Nardi, 1999). To design and manage organizations
is therefore crucial to understand the nature and structure of invisible work. Implicit information, e.g. the spatial location and the physical appearance of the objects of interest, is often significant only to users who possess the knowledge to interpret it. Most of this knowledge is not made explicit and codified but is tacit knowledge, namely it is knowledge that users possess and currently use to carry out tasks and to solve problems, but that they are unable to express in verbal terms and that they may even be unaware of. It is a common experience that in many application fields users exploit mainly their tacit knowledge, since they are often more able to do than to explain what they do. Thus, as suggested in (Karasti, 2001), “to make the invisible visible, to tease out the implicit, complex workplaces need to be investigated for the practical task-oriented activities”. This implies the need of a close observation of everyday work practice settings, rather than simply interviewing practitioners about their working activities.

3. A system acceptable by its users should have a gentle slope of complexity: this means it should avoid big steps in complexity and keep a reasonable trade-off between ease-of-use and functional complexity. Systems might offer users, for example, different levels of complexities in performing EUD activities, going from simply setting parameters, to integrating existing components, up to extending the system by developing new components (Myers et al., 1992; Myers et al., 2003). The system should then evolve with the users (co-evolution) (Arondi et al., 2002; Bourguin et al., 2001), thus offering them new functionalities when needed.

4. Co-evolution forces the design team in a continuous development of the system. This is carried out on one hand by end users, who can perform a tailoring activity adapting the software environments they use to their own needs and habits. On the other hand, end users should collaborate with all other stakeholders both in the design and in the evolution of the whole interactive system.

3. SSW methodology

Starting from the above observations, three principles are at the basis of our methodology to design interactive software systems: i) the language in which the interaction with systems is expressed must be based on notations traditionally adopted in the domain; ii) systems must present all and only the tools necessary to perform the user work, without overwhelming users by unnecessary tools and information; iii) systems must provide a layout conveying implicit information by simulating the traditional layout of the tools employed in the domain, such as for example mechanical machines or paper-based tools.
Our methodology emphasizes a perspective on meta-design that goes beyond, but includes User Centered Design (UCD) and participatory design: UCD relies on a deep study of the end users, their tasks, and their involvement in prototypes and system evaluation; participatory design means that representatives of end users participate in the design team. In our view, meta-design means that design environments are developed in a participatory way and provided to end users, permitting them to shape their application environments. Thus, end users play two main roles in the lifecycle of the interactive software system: 1) they perform their working tasks; 2) they participate in the development of software environments as stakeholders of the domain.

In the first role, as users performing working activities, end users can tailor the software environment to their current needs and context. Practitioners, such as mechanical engineers, geologists, physicians, often work in a team to reach a common goal. The team might be composed by members of different sub-communities, each sub-community with different expertise. Our approach to the design of a software system in a certain domain is to see the system as composed of various environments, each one for a specific sub-community of end users. Such environments are organized in analogy with the artisans workshops, where the artisans find all and only the tools necessary to carry out their activities. Following the analogy, end users using a virtual workshop, called Software Shaping Workshop (SSW), find available all and only the tools required to develop their activities. A type of SSWs, which allow end users to perform their daily tasks, are said application workshops. End users may also perform EUD activities, for example by using annotation tools permitting the definition of new widgets (Carrara et al., 2002): as a reaction to the annotation activity performed by the end user, the workshop may transform the annotated document area into a new widget, to which a computational meaning is associated. This widget is added to the common knowledge base and becomes accessible by other end users, each one accessing the data through his/her own workshop enriched by the new widget that is adapted to the specific context.

In the second role, as members of the design team, end users participate directly in the development of the workshops for their daily work. In fact, even if they are non-professional software developers, are required to create or modify application workshops. To this end, different workshops (system workshops) are made available to them, which permit the customization of each application workshop to the end user community needs and requirements. The concept of system workshop is more general: actually, system workshops exist that allow the members of any community involved in design and validation of the
system to participate in this activity. For example, system workshops for Human-Computer Interaction (HCI) experts and software engineers are used.

Each member of the design team can examine, evaluate and modify an application workshop using tools shaped to his/her culture. In this way, this approach leads to a workshop network that tries to overcome the difference in language among the experts of the different disciplines (software engineering, HCI, application domain) who cooperate in developing computer systems customized to the needs of the user communities without requiring end users to become skilled in all the involved domains of knowledge.

The network is structured so that the different experts can participate in the application workshops design, implementation, and use without being disoriented. Every expert can access more than one workshop. For example, an end user as practitioner uses an application workshop, while as member of the design team (representative of the user community) uses a system workshop; a software engineer uses a particular system workshop to produce the tools for the other members of the team, and accesses other workshops to check their functionalities.

Figure 1. The SSW network. Dashed arrows indicate communication paths and full arrows indicate generation and evaluation paths.

In general, a network is organized in levels. In each level, one or more workshops can be used, which are connected by communication paths. In the example in Figure 1, three levels exist: a) the top level (the meta-design level), where software engineers use a system workshop, called W-SE, to prepare the tools to be used and to participate in the design,
implementation, and validation activities; b) the *design level*, where software engineers, HCI experts, and end user representatives cooperate to the design, implementation, and validation activities; a design member belonging to the community X participate in the design using a system workshop W-ReprX customized to the needs, culture and skills of community X; c) the *use level*, where practitioners of the different communities cooperate to achieve a task; similarly, practitioners belonging to the community X participate in the task achievement using the application workshop W-End-User-X customized to their needs, culture, and skills.

Software engineers are required: a) to provide the software tools necessary to the development of the overall application; b) to participate in the design of application and system workshops. Therefore, from their workshops software engineers may reach each system and/or application workshop.

Representatives of end users may work at two levels. At the design level, they use their own system workshop to participate in the design and, at the bottom level, they use their own application workshop to carry out specific activities in their application domain.

HCI experts use their own system workshop to participate in the design and, at the bottom level, may access all application workshops to check their functionalities.

Practitioners can only use their own application workshop to perform their tasks.

At each level, communication paths exist, which allow experts in a domain to communicate with experts in a different domain. An expert using his/her workshop W-ZZ can send data or programs to a different expert. Data and programs are interpreted and materialized by the workshop W-UU, customized to the second expert. As we have shown in (Costabile et al., 2005), the main tool currently used for this kind of communication is electronic annotation.

It is important to notice that a communication exists from a lower level to the upper one and viceversa. In our approach, this capability is given by: a) allowing end users, interacting with an SSW, to annotate their usability problems and to communicate them to all the experts reachable in the network; b) allowing designers to update the applications and propose the updated versions to end users.

Each SSW co-evolves in time with end users and the design team. Co-evolution is a long life process asking for a *continuous development* of the interactive system. This is different from the so-called Rapid Application Development (RAD) that foresees the possibility for programmers to quickly build working programs. RAD systems emphasize reducing development time, while the SSW focuses on user satisfaction, i.e. 1) acceptability and usability of the application, and 2) balancing the features of the tools on user needs and
expectations. Moreover, RAD systems are development environments devoted to software engineers, but they are still far to be used easily and effectively by end users. The SSW approach stresses the role of representative of end users as active members of the design team. To this end, system workshops at design level, used by either HCI experts or representatives of end users, are customized to the user community to which they are devoted, speak user languages and present all and only the tools necessary in that context for developing further workshops.

4. Refining the concept of meta-design

Meta-design emerges from the practice of Computer Science, in particular from what is now called End-User Development (Lieberman et al. 2005). Recently a definition has been given in (Fischer et al., 2004; Fisher and Giaccardi, 2005): “meta-design characterizes objectives, techniques, and processes for creating new media and environments allowing ‘owners of problems’ (that is, end users) to act as designers. A fundamental objective of meta-design is to create socio-technical environments that empower users to engage actively in the continuous development of systems rather than being restricted to the use of existing systems”. We refine and clarify here this definition on the basis of our experimental activities.

End users are indeed the “owners of problems”, and have a domain-oriented view of the processes to be automatized. Moreover, they are not expert in HCI nor in software engineering, so they can act as designers contributing their experience on the domain of activity. In turn, software engineers have the knowledge about tools and techniques for system development and the HCI experts have the knowledge on system usability and human behaviour. HCI experts and software engineers are stakeholders whose contribution is necessary to the development of the system because they are the only ones who can guarantee the usability and the performance of the system.

The SSW methodology offers to each stakeholder a software environment, a Software Shaping Workshop, by which the stakeholder can test and study a software artefact and contribute to its shaping and reshaping as any object or tool that can be easily created, manipulated and modified. HCI experts, software engineers, and users acting as developers, each one through his/her SSW, can access, test, and modify the system of interest according to his/her own culture, experience, needs, skills. They can also exchange the results of these activities to converge to a common design.
In the light of these considerations, meta-design is a technique which provides the stakeholders in the design team with suitable languages and tools to favour their personal and common reasoning about, and collaboration to, the development of software systems that support user work.

End users must be in the situation to act as designers when they need and to act as end users when the tools match their needs. This two-fold roles of end users is discussed also by Fischer in (Fischer, 2002), where he argues about the “consumer” and “designer” perspectives by saying “that the same person is and wants to be a consumer in some situations and in others a designer; therefore ‘consumer/designer’ is not an attribute of a person, but of a context”.

On the whole, the development of a system to support the work practice in a specific domain of application results into the development of a network of system and application workshops. The design team is engaged in a continuous development of the system. This continuous development is carried out on one hand by end users of application workshops, who can perform a tailoring activity adapting the application workshops they use to their own needs and habits; on the other hands, all other stakeholders participate in system evolution, by customizing the application workshops using their system workshops.

In this organization, the personalization activities of application workshops can be performed by the design team or by the end users themselves. Thus we classify personalization activities into customization and tailoring. Customization is the activity performed by the design team which generates application workshops for a specific community of users by exploiting users’ notations, dialects, principles, and standard rules. Tailoring is the activity performed by end users to adapt an application workshop to the current activity and context of work. The idea is to permit tailoring of systems, which are already specific and suitable to the needs of a specific community of users, thus allowing users a further step of individual personalization. We call this activity tailoring toward the individual (or individual tailoring), which is performed by the users through small incremental steps.

Different types of individual tailoring can be devised: tailoring for individual work, concerning the activities that the specialist can perform to adapt her/his environment during her/his own work; tailoring for cooperative work, including those activities that the specialist performs to prepare the information that will be provided to another specialist to whom a consultation is requested.
5. SSWs in the medical domain

In this section we describe a project in the medical domain, to which we are currently applying the SSW methodology.

The improvement of the quality of the medical diagnosis is the main goal of each physician. Thanks to the evolution of research and technology in the medical domain, each specialist may have the aid of medical examinations of different types, i.e. laboratory examinations, X-rays, MRI (Magnetic Resonance Imaging), etc. A team of physicians with different specializations should analyze the medical examinations giving their own contribution according to their “expertise”. However, the increasing number of diagnostic tools and medical specializations as well as the increasing number of patients do not permit the team of specialists to meet as frequently as needed to analyze the clinical cases especially if they do not work in the same building or moreover they work in different towns or states.

The information technology has today the potential of overcoming this difficulty by providing software environments that allow a synchronous and/or asynchronous collaboration “at a distance”. Thus, the specialists do not need to meet at the same moment for analyzing the clinical cases on which they collaborate. Software environments will give the possibility to each specialist to analyze the medical cases of different patients and to formulate her/his own diagnosis taking into account the opinions of the other colleagues without the need of a synchronous consultation.

There are already tools for supporting the physicians to formulate the medical diagnosis, e.g. telemedicine, videoconference, etc. They are very sophisticated and often they need large system resources. Moreover, physicians complain that although these tools are very expensive, they are designed more for computer experts than for physicians. In the experience with physicians collaborating with us, these tools present personalization features very difficult to be learned and used by them. Our proposal of SSWs aims at providing the physicians with software tools that are first of all easy to use and adequate to the physicians’ current tasks. This would determine an increase in end user productivity and performance, with the achievement of competitive advantage for the organization they work in, by permitting consultations among physicians without constraints of time and place.

In the case study with radiologists and pneumologists (Costabile et al., 2005), we adopted the SSW approach to provide physicians with software tools which are both usable and tailorable to their needs. The case study had the goal to support different communities of physicians, namely radiologists and pneumologists, in the analysis of chest radiographies and in the generation of the diagnosis. Radiologists and pneumologists represent two sub-
communities of the physicians community: they share patient-related data archives, some models for their interpretation, some notations for their presentation, but they have also to perform different tasks, documented through different sub-notations and tools. Therefore, their notations can be considered two (visual) dialects of the physicians’ general notation. As a consequence, two different application workshops have been designed for these two communities of users. The pneumologist and the radiologist involved in the study of the pulmonary diseases, even if they are working in different wards or different hospitals, can define an agreed diagnosis using each one her/his application workshop tailored to her/his culture, skills, and articulatory abilities in an asynchronous and distributed way.

We are currently applying the same approach in a larger project we are involved, which is in collaboration with the physicians of the neurology department of the “Giovanni XXIII” Paediatric Hospital of Bari. In this project, different communities of physicians are involved, namely neurologists and neuro-radiologists, in the analysis of clinical cases and in the generation of the diagnosis. Neurologists need to exchange consultations with other neurologists and/or neuro-radiologists in order to make a better diagnosis for their children patients. These groups of physicians autonomously organized a procedure for the exchange of information (data, images and text), using common network tools such as e-mail. However, the physicians were not satisfied with the quality and reliability of such procedure, so that we started a collaboration with them with the objective of creating software environments that might satisfy their needs.

We developed a first version of the prototypes devoted to neurologists and neuro-radiologists, on the basis of the experience gained in the previous project with radiologists and pneumologists (Costabile et al., 2005). Then, usability evaluation of these prototypes was performed by applying different techniques, such as heuristic evaluation, cognitive walkthrough and user testing. In parallel, we carried out an accurate field study in the neurology department of the hospital and we developed a second version of the prototypes. In particular, the field study was aimed at understanding and identifying the environmental and organization factors that influence the work of the physicians and the flow of activities, as well as the communities of end-users involved and their main tasks, their common languages, their specialized medical dialects. Various methods have been considered to perform this study, such as user observations, interviews, analysis of users’ documents and languages. From our previous experience with this particular community of users, we decided to perform the field study to deeply know the work practice of the physicians.
Prototypes of SSWs have been developed on the basis of the information collected during the field study. As discussed in Section 3, the resulting interactive system is structured as a network of SSWs, each specific for a community of users. Being the network modular, we foresee the possibility in the future to extend it by creating other SSWs for other stakeholders, e.g. clerks and managers dealing with management and billing systems.

5.1 The field study

The field study was mainly aimed at understanding how the physicians collaborate in the analysis of clinical cases. In accordance to the basic principle of our methodology (mentioned in Section 3), we also wanted to look at: a) the notations adopted in the specific domain, b) the documents the various users exchange, c) the tools they usually use.

During the field study, the analysts periodically observed the physicians during their daily work in the hospital (about 2-3 visits per month since 2003). They observed meetings of physicians of the same departments and meetings of physicians with different specialization. Sometimes, they performed semi-structured interviews for better understanding documents, tools and languages. The information collected during the study has been used to identify the right requirements of the application.

The stakeholders identified through the field study are: the neurologist; the neuroradiologist; the patient, who is the child with neurological troubles and her/his family; the family doctor, who knows the symptomatology of the patient and prepares a diagnostic question for the neurologist; the internal laboratory, which performs the examinations prescribed by the neurologist and is sited in the hospital in which the neurologist works; the external laboratory, which also performs examinations but is outside the hospital in which the neurologist works.

Usually, a patient with troubles first goes to her/his family doctor. The doctor suggests to the patient family to go to a neurologist and prepares a diagnostic question for her/him. The neurologist studies the symptomatology of the patient and prescribes others medical analyses, that can be performed in the internal laboratory, such as EEG (electroencephalogram), or in the external laboratory, such as blood test, MRI and so on. When the neurologist has the analysis results of the patient, s/he studies them accurately and often identifies critical results and makes some annotations on them. Then, s/he defines a diagnosis and gives it to the patient or her/his family. In particularly serious and difficult cases, before formulating a diagnosis, the neurologist needs to ask other physicians for a consultation to better reason on the pathology of the patient. For example, the neurologist
can decide to refer to a neuro-radiologist for a more detailed analysis through MRI. The neuro-radiologist provides her/his opinion to the neurologist. Both neurologist and neuro-radiologist can also ask for a consultation to another colleague, neurologist or neuro-radiologist respectively, who is specialized in particular pathologies. At the end of the consultations, the neurologist provides the diagnosis and the treatment to the patient. The information flow just described is represented in Figure 2.

In our project, we have first focused on the interaction that occurs among neurologists and neuro-radiologists who collaborate in formulating a diagnosis. We now describe a scenario in which a neurologist asks for a consultation to a neuro-radiologist to analyze clinical cases by studying MRI. In our scenario, both physicians are male.

Currently, consultations occur during a real meeting. Due to the busy schedule of the physicians, these consultations cannot be frequent, therefore when they meet they have to analyze several clinical cases. During the meeting, the cases are discussed one at a time and always with the same procedure. The neurologist chooses a case, gives the MRI plats to the neuro-radiologist and begins to tell the most relevant data about the patient history. The neuro-radiologist puts 3 or 4 MRI plats on the diaphanoscope and begins to study them (Figure 3).
During the study of the MRI, neurologist and neuro-radiologist exchange information in order to clarify possible doubts and converge to an agreed opinion (Figure 4). At the end, the diagnosis on which the specialists agreed is written on the patient record and the next clinical case is considered.

If the physicians cannot physically meet, the consultations occur through e-mail. Obviously, in this case some problems arise, such as limited capacity of the e-mail, connection problems and so on.
The analysis of the information collected during the field study allowed us to develop the prototypes described in the next section.

5.2 System customization and tailoring

The described scenario is a typical case of cooperative work. We adopted the SSW methodology to build application workshops customized to the physicians’ notation, language, culture, and background, that physicians themselves can further tailor according to their needs. These workshops allow the specialists to cooperate in virtual meetings. The specialists may use their own application workshops to perform their working tasks: for example, a specialist may analyze the available EEG or MRI, perform annotations and/or computations on them, select parts of them, define diagnoses and/or consultation requests.

The observations collected during the field study are at the basis of the design of the application workshops to be used by the two different communities of physicians. Differently from the case of the pneumologists and radiologists (Costabile et al., 2005), the application workshops designed for the physicians of the new project (neurologists and neuro-radiologists) have an overview area on the top of the screen which may be used to browse MRI plats or EEG portions. The overview area is the electronic counterpart of the diaphanoscope used by the physicians in a real meeting (see Figure 5). During the observations, we noticed that neuro-radiologists are only interested in MRI; thus, in their application workshop, they find only the MRI overview area (see Figure 5) together with tools to process the MRI and to formulate the diagnosis.

Figure 5. The application workshop prototype devoted to the neuro-radiologist.
On the other hand, neurologists study primarily a great number of EEGs but in some cases, they analyze MRI plats. Thus, in their application workshop there are two overview areas, which are resizable (see Figure 6). In this way, the neurologists can reduce (or even close completely) the area containing the MRI plats, in order to expand the EEG overview area according to their needs.

Figure 6. The application workshop prototype devoted to the neurologist.

Each workshop is designed to be equipped with a certain set of tools. Some are provided as default. If the physician should need to eliminate or add some tools, the workshop permits this. Adding or deleting tools, resizing the overview area are examples of tailoring for individual work activities.

As already mentioned in Section 4, tailoring for cooperative work refers to tailoring activities needed to communicate efficiently and effectively with colleagues, for example to shape software artefacts made available to another specialist to whom a consultation is requested. In the following, we describe how a neurologist may tailor the system in order to communicate with a neuro-radiologist and request her/him a consultation.

As the field study revealed, in their daily work, neurologists and neuro-radiologists highlight parts of EEG and/or MRI. As shown in Figure 6, in the neurologist application workshop, a portion of EEG can be selected from the EEG overview area and automatically loaded in a specific window, called working bench, which appears in the working area. The working bench is equipped with a toolbar hosting the tools to be used by the neurologist to study the EEG and prepare requests of consultation to be sent to other specialists. According to the principles at the basis of SSW methodology, these tools resemble the real tools the physicians use in their work practice.
Moreover, neurologists and neuro-radiologists request and provide consultations by indicating to their colleagues parts of EEG and/or MRI which may be of interest to formulate the diagnosis. This kind of activity is supported by the application workshops we have developed. For example, the neurologist may find in the working bench two different tools for selecting limited areas in the portion of EEG: 1) one allows the physician to circle the area of interest; 2) the other permits to identify, through two vertical red bars, a limited part of the selected portion of EEG concerning a certain period of time (see Figure 6).

A selected area can be annotated and the annotation exploited to support the collaboration between specialists that want to reach a common diagnosis. For example, if the neurologist needs to consult a neuro-radiologist, he makes a request by opening a special type of annotation window, called *consulting window*, shown in Figure 7. This window permits to articulate the annotation into two parts: the *question* to be asked to the colleague; and the *description* which summarizes information associated to the question. A third part can be tailored according to the addressee of the consultation request: if s/he is a physician who needs more details about the clinical case, the sender may activate the *detailed description* and fill it, otherwise he can hide it.

![Consulting window](image)

Figure 7. An example of consulting window.

In other words, the physician who wants to ask for a consultation is allowed to prepare a tailored annotation specific to the physician he is consulting. In a similar way, a physician can make a different type of annotation in order to add a comment that can be stored and possibly viewed by his colleague. It is worth noting that both types annotation never damages the original image.

Back to the example of a consulting request of the neurologist for the neuro-radiologist, the neurologist can save the annotation, that can be successively viewed by his
colleagues. When the neuro-radiologist starts working with his application workshop and finds the request of the neurologist, he reads all the information and answers by filling the proper fields in the consulting window (see Figure 8).

6. Related work

In 1993, Brancheau and Brown describe *end user computing* as "… the adoption and use of information technology by people outside the information system department, to develop software applications in support of organizational tasks" (Brancheau and Brown 1993). The organization in which such people work requires them to perform end user computing and to assume the responsibility of the results of this activity. Indeed, one fundamental challenge is to develop environments that allow people without a particular background in programming to develop and tailor their own applications. In this line, we foresee the active participation of end users in the software development process.

Traditional participatory design approaches exploit techniques derived from social theories that support communication and collaboration within the interdisciplinary team: such techniques move from just system descriptions, to collaborative construction of mock-ups, cooperative prototyping, game-like design sessions. Most of these techniques not only analyze solutions after setting up goals, but also use fantasy and imagined futures to study specific actions (Bodker, Gronnbaek, Kyng 1993). In (Greenbaum and Kyng 1991) the Future Workshop technique is discussed: it foresees group meetings run by at least two
facilitators and having the aim of analysing common problematic situations, generate visions about the future and discuss how to realize these visions. In our approach, the term “workshop” is used with a different meaning, precisely that one provided in every English dictionary: e.g. “a small establishment where manufacturing or handicrafts are carried” (Merrian-Webster online). We speak about “Software Shaping Workshops”, referring to software environments that enable users to “shape” virtual tools and data. In other words, the real workshop is the metaphor we adopt for the conceptual model of our software environments. Participatory design of visual interactive systems is performed through the interaction with SSWs customized to the needs, preferences, habits and capabilities of each specific community of users belonging to the interdisciplinary team. A similar approach in participatory design literature is Cooperative Prototyping (Bodker and Gronbaek 1991), in which prototyping is viewed as a cooperative activity between users and designers. Prototypes are developed by software engineers, then discussed with users, and possibly experienced by them in work-like situations. Prototype modifications may be immediately made by direct manipulation, also by users, during each session of participatory design. However, in such approach, prototypes just represent an interactive digital evolution of paper-based mock-ups: real systems are then re-programmed and all modifications require large programming effort that are postponed and made by designers after each session. Our approach foresees the participation of all the stakeholders in the development of the final system, each one according to their own view, through the use of SSWs. Therefore, our SSWs can be regarded as a new technique supporting participatory design.

Fisher et al. propose SER (Seeding, Evolutionary growth, Reseeding), a process model to design systems as seeds, with a subsequent evolutionary growth, followed by a reseeding phase; it is used for the development and evolution of the so-called DODEs (Domain-Oriented Design Environments), which are “software systems that support design activities within particular domains and that are built specifically to evolve” (Fischer et al., 2001). Three intertwined levels of design activities and system development are envisaged: at the lower level, a multifaceted domain-independent architecture is present constituting the framework for building evolvable systems; at the middle level, the multifaceted architecture is instantiated for a particular domain in order to create a DODE; at the top level, individual artefacts in the domain are present, being developed by exploiting the information contained in the DODE. In (Fischer et al., 2004) meta-design is proposed as an evolution of DODEs and “a vision in which design, learning and development become everyday working practice”. The SER model is still adopted to support meta-design. Our SSW approach has
some similarities with this work, but it emphasizes the need of providing personalized environments to all stakeholders, in terms of language, notation, layout, interaction possibilities.

In our approach, end users play a role similar to handymen in (MacLean et al., 1990). The handyman bridges between workers (people using a computer application) and computer professionals; s/he is able to work alongside users and communicate their needs to programmers. Similarly, representatives of end users bridge between workers and computer professionals, but are end users themselves. To participate in SSWs development, they must be provided with environments that are adapted to their culture, skills, and articulatory abilities. Besides the projects in the medical domain already discussed in this paper, in (Costabile et al., 2004) we describe an environment devoted to mechanical engineers who were the representatives of end users involved in the development of the application workshop devoted to assembly-line operators.

Karasti (2001) explores the integration of work practice and system design and insists on increasing the sensitivity of system design towards everyday work practice. She characterizes work practice by describing the complex social organisation, technological mediation, knowledge and meaning as socially constructed, and the intertwined nature of the unfolding activities in which all these aspects are joined. Everyday work practice has historically been invisible in design. We agree that an understanding of current work practice is useful in the design of new technologies. Moreover, two different bodies of knowledge are explored in (Karasti, 2001) to make work practice visible and intelligible for system design: the actual work activities and knowledge of practitioners, and what is considered relevant information for requirements analysis in system design. Thus, the challenge is to dissolve the barriers existing between designer and user knowledge, and to search for adequate methods to secure the inclusion of practitioner and work practice knowledge in design.

Research with similar interests in work practice has been carried out in studies that intertwine work practice and system design, especially within the field of Computer Support Cooperative Work (CSCW) (e.g. Suchman et al., 1999; Suchman and Trigg, 1991; Goodwin and Goodwin, 1997; Hughes et al., 1992), and can also be found in collections such as (Greenbaum and Kyng, 1991; Chaiklin and Lave, 1993; Button, 1993; Engeström and Middleton, 1996; Resnick et al., 1997).

In (Stary, 2000; Penner and Steinmetz, 2002) task-based and user-centered development approaches are presented to support the automation of user interface design. TADEUS project (Stary, 2000) proposes a development methodology starting from a
business intelligence model to generate user interfaces or portals by integrating a model-driven, task-based, user-oriented, and object-driven life cycle. Moreover, to cope with the complex and continuously changing needs of end users, Penner and Steinmetz propose an iteratively created operational prototype called DIGBE (Dynamic Interaction Generation for Building Environment) (Penner and Steinmetz, 2002). DIGBE is a multi-platform and a generic building control system (heating, cooling, ventilation, access control, security, and so on). An expert user, typically a building manager, starts up DIGBE application and, through a simple dialog, sets the initial state of the system. In response, DIGBE creates a dedicated child application for the other expert users (i.e. managers, operators, technicians), accessible through a log-on screen. After the expert user successfully enters the new system, DIGBE designs and presents (in real time) a user interface dedicated to the underline task set of the logged user. Moreover, during the interaction, the system dynamically adapts the ongoing user interface. This methodology is similar to the SSW methodology, but it is simply user-centered and leads to a unique general suitable system that can be specialized according to the domain and user community needs.

Finally, other works focus on experience-centered domains, that is “domains requiring six to 12 years of intensive practice before practitioners achieve the most effective levels of skill” (Hayes, 1985). In these domains (i.e. medical diagnosis, chess, professional design, planning tasks, etc.), one of the main challenges in building decision support is that users, at different levels of domain experience, have often very different needs. For example, a system designed to satisfy experts specific needs may frustrate novices and vice-versa. DAISY (Design Aid for Intelligent Support System) (Brodie and Hayes, 2002) is a design methodology for building decision support systems in complex, experience-centered domains. It provides a technique for identifying the specialized needs of users within a specific range of domain experience.

7. Conclusions

In this paper, we have illustrated how the SSW methodology to design software environments that support work practice and EUD activities is applied to a project in the medical domain. This methodology offers to each stakeholder a software environment, called Software Shaping Workshop, by which the stakeholder contributes to shape software artefacts. In this way, software engineers, HCI experts, end users acting as developers, each one through his/her SSW, can access and modify the system of interest according to his/her own culture, experience, needs, skills; they can also exchange the results of these activities
to converge to a common design. The proposed approach fosters the collaboration among communities of end users, managers, and designers, with the aim of increasing motivation and reducing cognitive and organizational cost, thus providing a significant contribution to the EUD evolution, as suggested in (Fisher et al., 2004).

We have discussed how the SSW methodology supports meta-design by allowing representatives of end users to be involved in the design of the workshops that will be used by all the end users belonging to the specific communities involved in the project. End users have then a twofold role: users and designers of their own software environment.

Moreover, the work described introduces a perspective on system personalization, distinguishing between customization and tailoring of software environments. The software environments are customized by the design team to the work context, culture, experience, and skills of the user communities; they are also tailorable by end users at runtime in order to adapt them to the specific work situation and users preferences and habits. End User Development and End User Computing are thus sustained by these two kinds of personalization.

The application of the SSW methodology to a project in a medical context has been described. Physicians and their activities have been carefully examined through a field study, in order to carefully understand their work practice and integrate it in the software design. Physicians are collaborating with enthusiasm to the development of the SSW prototypes. They understand and appreciate the novel approach of being involved in collaborative design processes, through which they can have a more active role than simple consumers of new technologies. Actually, available off-the-shelf software in the medical domain are designed to support specialized activities, such as image processing, clinical data organization, and statistical analysis. However, they do not support physicians in integrating these activities for diagnostic purpose. SSWs permit end users to carry out the activities that physicians are used to perform face-to-face (e.g. exchanging consultations) and that off-the-shelf software does not address yet. We are confident that this approach may determine an increase in end user productivity and performance, i.e. a better quality of diagnosis and medical cure.

8. Acknowledgments

The authors wish to thank the physicians of the Hospital “Giovanni XXIII” in Bari for their collaboration. We are also grateful to Giuseppe Fresta for his contribution to the implementation of the prototypes presented in the paper.

The financial support of Italian MIUR is acknowledged.
References


*Proc. Conference on Computer-Supported Cooperative Work (CSCW’92)*, ACM Press, 
Toronto, 115-122.


Merriam-Webster online, http://www.m-w.com/.


