Journal of Visual Languages and Computing 23 (2012) 78–90

Contents lists available at SciVerse ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

End users as co-designers of their own tools and products ☆, ☆☆

Carmelo Ardito *, Paolo Buono, Maria Francesca Costabile, Rosa Lanzilotti, Antonio Piccinno

Università di Bari Aldo Moro, Dipartimento di Informatica, via Orabona 4, 70125 Bari, Italy

ARTICLE INFO

Available online 18 November 2011

Keywords:
End-user development
Meta-design
Design model

ABSTRACT

In our Age of exponential technological advance, recent developments are determining an evolution of end users from passive information consumers into information producers. Users are increasingly willing and, indeed, determined to shape the software they use to tailor it to their own needs. Based on a brief review of research activities we performed in the last decade, this paper analyzes some challenges that software designers face to comply with the new roles of end users in the software life cycle, and discusses how to provide end users with software environments that empower them to become co-designers of their own tools and products. The examples reported in the paper show why and how end users are involved in design activities in various application domains.

1. Introduction

Software developers and users were traditionally considered as two distinct communities: developers would create computer systems that would then be used by users to perform their daily activities. By contrast, in today's information and communication society, more and more people do not only use software but also get involved in creating or modifying it. Thus, their role is no longer that of passive consumers of computer tools, but is evolving toward a more active role of information and software artifacts producers [1]. Web Content Management Systems and Enterprise Resource Planners are significant examples in which users are willing to shape software [2–7], but this is true in many other domains, as will be shown in this paper.

Even if the two terms *user* and *end user* will be used as synonyms in this paper, except in cases of ambiguity, it is worth distinguishing between them: *users* are all people interacting with software systems, so even software engineers are users of the tools and environments (e.g. CASE tools) they use for their work; *end users* are people who are not experts in Computer Science, nor willing to be, but who use computer systems for their daily activities, for work as well as for entertainment or other purposes [8]. Recent technology and the rise of the so-called Web 2.0 now offer end users the possibility to perform various activities that involve modification or even creation of software artifacts: they range from

☆ *Dedication*: This paper is dedicated to the memory of Piero Mussio. It refers to some research developed by our research group at the University of Bari working with Piero and his collaborators at the University of Brescia and the University of Milan. It also refers to some very early papers by Piero, which show that he was one of the pioneers in visual languages, in Human–Computer Interaction, and in end-user development. Since the 1970s, when Human–Computer Interaction had not yet become a discipline in its own right, Piero devoted close attention to user needs in the interaction with databases, pointing out important issues such as the diversity of users of interactive systems, the need for computer scientists to develop software environments and tools that take into account languages and notations adopted by users in their work practices, the possibility to allow users to adapt and refine by themselves the environments and tools they use. In the years we collaborated with Piero, we learnt to appreciate his wide culture, his scientific and human qualities; we argued a lot in many discussions about our work, and we shared some very enjoyable moments of our lives.

☆☆ This paper has been recommended for acceptance by S. Levialdi.

* Corresponding author.
E-mail addresses: ardito@di.uniba.it (C. Ardito), buono@di.uniba.it (P. Buono), costabile@di.uniba.it (M.F. Costabile), lanzilotti@di.uniba.it (R. Lanzilotti), piccinno@di.uniba.it (A. Piccinno).

1045-926X/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2011.11.005

simple parameter customization to more complex activities, such as variation and assembling of components [9,10]. These activities are examples of End-User Development (EUD), as defined in [11]. In short, end users are becoming co-designers of their software tools and services [12]. This evolution poses a challenge to professional designers (software engineers), who have to create software environments that can empower end users to shape the software they use, without obliging them to become programmers.

This paper starts with a brief discussion of the roles of end users in the software life cycle, and describes joint research work performed with Piero Mussio and some of his collaborators at the University of Brescia and the University of Milan over the last decade. One of Piero Mussio's major concerns was the diversity of end users, this also influenced the research reported in this paper. Based on various experiences about the development of interactive systems that address people's activities in their daily practice, a design approach has been defined and progressively refined [8,13,14]. The aim is to create systems that permit EUD, i.e. that support people to tailor the software they use to make it better suited to their own needs, also by creating or modifying software artifacts. EUD is not a luxury but a necessity [1], demanding new paradigms and approaches to create systems to comply with the new roles of end users in the software life cycle. This paper describes one of such approaches and discusses its application through examples that show, in various contexts, how end users become co-designers of software.

The paper is organized as follows. Section 2 briefly discusses how end users roles are evolving towards a more active participation in the development of computational artifacts. Section 3 illustrates some challenges faced when addressing the diversity of end users. Section 4 describes how end user involvement in the whole software life cycle and the need to support EUD are determining a shift toward a new software design paradigm; the approach developed, based on meta-design, is briefly reported. Section 5 presents some examples, which show how our approach to create systems enabling end users to become co-designers of their tools and products is adapted to comply with different application domains. Section 6 concludes the paper.

2. About end users

The desire of end users to adapt software to their needs had already begun to be analyzed by various authors by the beginning of 1990s; empirical studies on the activities end users were willing to perform were reported in [15,16]. End-User Development and End-Users Software Engineering are terms that have been coined in the last years to refer to these activities [11,17–19]. End users are not pure end users any more, but are determined to become increasingly involved in software development; some may have certain software development skills but, surely, they are not interested in software per se, they simply want to solve their own specific problems [14].

The activities performed by end users, which result in software modification, were analyzed in [20]. Two main types were identified: *parameterization* and *tailoring*. *Parameterization* refers to activities that allow end users to choose among alternative behaviors (or presentations or interaction mechanisms), already available in the application, by setting some parameters. Other names used in literature for such activities are customization or personalization (see, for example [11]); they are already possible in widely used applications such as Microsoft Word™. *Tailoring* includes all activities that involve some programming in any programming paradigm, thus creating or modifying a software artifact. Other analyses of end-users activities are reported in [21,22].

Our research was originally driven by the specific needs of people we worked with, who were experts in a specific discipline, such as geology, mechanical engineering, medicine, etc.; they required our expertise to develop interactive systems to support their daily activities. We called these end users *domain-expert users*: they are experts in a specific domain but not necessarily experts in Computer Science [13]. Today, thanks to the increasing availability of technology and software applications, we are facing an exponential increase of the number and type of end users, ranging from very young to elderly people, who use computer applications on the Web and on various mobile devices to support their daily activities. These are not confined to work, but also include information needs, or entertainment and pleasure purposes. A study published in 2005 estimated that in 2012 there will be 90 million end users in American workplaces and fewer than 3 million professional programmers [23].

Most end users have no programming skills and do not want to be constrained by formalisms unfamiliar to their culture. They wish to use software environments that are easily accessible and usable, but which they can tailor to their needs, tasks and habits. Even if they use software applications that allow them to create or modify software artifacts, for example by creating a macro in Microsoft Excel™, they do this without being aware that what they are doing is programming. In other words, they are *unwitting programmers* [24], like the children analyzed in [25]. Indeed, children playing with a computer often use sophisticated programming skills, but this is embedded in an intrinsically motivated activity that they perceive as something easy and fun to perform. They do not regard it as programming which, in the child's opinion, is very difficult. Various computer applications allow children to construct interactive simulations, animations, and games, in a manner that places a lot of emphasis on construction; they program by composition, not by algorithm development: their intention is to play and enjoy, and they do not even realize they are programming. Children learn by trying things out. When they use a new environment, they do not read tutorials, but go straight to the example or ask friends. Children greatly enjoy communicating with friends and possibly performing collaborative activities, often conducted remotely by exchanging artifacts online. This behavior is very similar to that of the end users we analyzed [8,14]: end users want to manipulate and tailor objects in their software

environments in order to create new configurations or new designs. They want to do this as a part of activities that they are highly motivated to perform, without thinking that they are programming. As a consequence, designers should create systems that empower end users by providing such communication and collaboration capabilities. This is a major point addressed by the EUD research community, as discussed in [26].

3. User diversity

Addressing user diversity has been a concern of our research since the very beginning. In particular, we observed that even the end users of the same interactive system are often diverse, constituting different communities characterized by specific cultures, goals, tasks and context of activity [8]. The projects reported in Section 5 provide various examples of diversity within the target end users of an interactive system. To ensure a good design, it is of critical importance to address this diversity of end users [27]. The slogan "one size fits all" does not work for user interfaces, as different users of a same interactive system may need different interfaces that provide them adequate support. It is well known that people experience many difficulties when they interact with a system that has been designed without taking into account their cultural background, their reasoning strategies, the way they carry out their tasks in their daily practices, the languages and notations they are familiar with.

Piero Mussio highlighted this problem in his very first works dating back to the 1970s. At that time, he was involved in satellite image processing, but his attention was already devoted to allowing domain-experts to interact in the best way with the databases storing such images, while carrying out their analysis work. Citing from one of his papers [28]: "The data related with the surface water system are utilized by several users, according to the interdisciplinarity of the system analysis… Specialists of different disciplines use their languages to describe phenomena and data… The same elementary objects and names appear in both descriptions but with different meaning and associated computations… The communication with the user must be different according to the various disciplines". In another section of that paper, he points out that scientists of different disciplines have developed their own models and languages, thus software tools have to match these languages.

It is amazing to see that well before the development of the worldwide Human–Computer Interaction literature, Mussio had a clear vision of how to support people interacting with computer systems by providing them with user interfaces capable "to speak the user language". He also declared that what the "information analyst" is required to do is to identify what can be coded in the user language (i.e. the language for the human–computer dialog), that is, the user is asked to design with the "information analyst" an "intermediate language" which is the basis of the query language for the specialized database [28]. Thus, he anticipates the importance of defining a user interface and an interaction language to mediate the communication with the database query language. It is worth mentioning that Mussio later became an active contributor to the field of visual languages, which appeared in the 1980s with the aim of defining easier to use formal languages. More importantly for the specific topic addressed in this paper, he foresaw the collaboration between users, as domain-experts, and professional developers (called "information analysts") in the design of the formal language to be used in the human–computer dialog. In other words, he already had a vision of end users being co-designers of their environments and tools, as the recent EUD approaches state.

User diversity has been addressed in our approach to interactive system design, which states that an interactive system should propose different interaction environments, each one suited to a specific community of users and adopting a language for the human–computer dialog that is inspired by languages and notations used in the community's daily pratice [8]. In a paper published in 1991, Mussio already recommended that "the variety of specialized visual and verbal expressions used by the expert in his traditional activity be translated into a set of computer languages and organized as an interactive environment" [29]. In another paper in 1992, he anticipated EUD and set the basis for the design model we later developed [30]; for example, he declared that, by collaborating with professional developers, the domain expert "specifies high-level visual languages by which he can program (i) the interaction necessary to execute and control his computational tasks, (ii) his own computing tools, (iii) how input data can be captured by the system in a way which is natural for the naïve user (naïve from the Computer Science point of view), and (iv) how output data can be presented in a form communicable to other experts". In summary, Piero Mussio had a very early vision that end users would actively participate to create their system, i.e. would need to be involved in the whole software life cycle. Our experience confirms this position, as discussed in the next section.

4. Involving end users in the whole software life cycle

The traditional life cycle of interactive systems distinguished between design time and use time. At design time, system developers would create environments and tools, figuring out end users' needs and objectives. At use time, end users would use the system. Design frameworks were based on the assumption that major design activities end at a certain point, after which the use time begins. End users would be active only at use time. Even when performing User-Centered Design, which requires the system to be designed by iterating a design–implementation–evaluation cycle, development was carried out by software professionals, while end users only use the system and, at most, are involved in prototype evaluation [31].

Participatory Design was introduced as a design paradigm which considered the participation of end users in the design process [32]. The rationale was that users are experts of the work domain so a system can be effective only if these experts are allowed to participate in its

design, indicating their needs and expectations. Thus, end users had become members of the design team, but no tools were yet provided to let them create or modify software.

More recently, EUD started the trend toward a more active involvement of end users in the overall software design, development, and evolution processes. Tasks that are traditionally performed by professional software developers are transferred to end users, who become co-designers of the tools and products they will use. Of course, end users have to be specifically supported in these new roles of designers and developers. This does not imply transferring the responsibility of good system design to them. It actually makes the work of professional developers even more difficult, since it is still their responsibility to ensure the quality of the artifacts created by end users. These issues are discussed in [18]. Some EUD-oriented techniques have already been adopted in software for the mass market, such as the adaptive menus in Microsoft Word™ or some Programming by Example techniques in Microsoft Excel™. iGoogle™ is another excellent example of a Web application which requires end users to perform EUD activities to tailor the available software tools to their own needs [33].

In order to allow end users to create and modify software artifacts, a two-phase design process must be considered: the first phase (meta-design phase) consists of designing the design environments, i.e. the environments suited to the diverse stakeholders who participate in the design of the final applications; the second phase consists of designing the final applications, using the design environments. The two phases are not clearly distinct, and are executed several times in an interleaved way because the design environments evolve, both as a consequence of the progressive insights the different stakeholders gain into the design process, and as a consequence of the feedbacks provided by end users working with the system in the field. This two-phase process requires another shift in the design paradigm, which moves from participatory design to meta-design (i.e. design for designers) [8,12].

In EUD approaches, the separation between design time and use time becomes fuzzy [34]. This is true also in recent software development methodologies, like Agile Development [35]. These two stages are now bridged into a unique "design-in-use" continuum that permits the creation of open and continuously evolvable systems, which are extended and/or redesigned at use time by end users collaborating with all the other stakeholders. This design paradigm considers software design as an evolutive and never-ending process, which can be modeled as a *design-develop-use-evolve* cycle. In agreement with other authors, we can say that the system is in a "perpetual beta" version [36].

The shift towards a new approach to create interactive systems was confirmed by a study commissioned by the U.S. Department of Defense, which clearly states that development activities will in the near future be distributed and initiated by various stakeholders [37]. Besides end users (of possibly different types), who "own" the problem, and software engineers, who "own" the technology, in the design team other experts and/or stakeholders are needed, e.g. Human–Computer Interaction (HCI) experts, who know human factors and may advise on the design of usable systems capable of generating a valuable user experience; marketing experts, who may advise on how to design a product with market appeal; graphic designers, who should contribute to the design of an attractive user interface; other experts of the system domain. These people contribute to the system design with their own expertise. They need different software environments, specific to their culture, knowledge and abilities, through which they can contribute to shape software artifacts. They should also be able to exchange among themselves the results of these activities in order to converge toward a common design.

Over the years, we have been working on the creation of software infrastructures that support EUD activities as well as knowledge creation and sharing among the stakeholders involved. This research resulted in the definition of a design approach based on the Software Shaping Workshop (SSW) model, which allows a team of experts to cooperate in the design, development, use and evolution of interactive systems [8,13,14]. The SSW model supports meta-design in that it prescribes that, instead of developing the final interactive system as in traditional design approaches, professional developers should design *software environments* for the different communities of stakeholders involved in the creation of the system, who will use such environments not only to carry out specific tasks at use time, but also to contribute to design and evolution of the interactive system [38]. These software environments are called *Software Shaping Workshops* (SSWs or briefly workshops). The term *workshop* comes from the analogy with an artisan's workshop (e.g. the joiner's or the smith's workshop), i.e. the workroom where the artisan finds all and only those tools necessary to carry out her/his activities. Each SSW provides an interaction language tailored to its users' culture, since it is defined by formalizing the traditional user notations and system of signs [39]. Communication channels among the various workshops are provided in order to support collaborative development and evolution. Thus, the workshops act as cultural mediators among the different stakeholders by presenting the shared knowledge according to their own languages. The SSWs are designed according to the "gentle slope of complexity" principle [10,27,40]: people find in their SSW only those tools and functionalities necessary to their tasks, presented in a way that is adequate to their culture and skills, so that they can easily use them. Of course, once users get familiar with their SSW, they may require new and more complex functionalities: such needs likely crop up later on determined by users' evolution during time.

To summarize the SSW approach, the first important step is a study aimed at accurately gathering users and system requirements. In particular, those stakeholders who may contribute to one or more of the phases of the software life cycle have to be identified and analyzed. Then, the meta-design team is defined which, besides professional developers who are the technology experts, and HCI experts, includes at least the domain and

problem experts. This team creates the SSWs for the involved stakeholders. Since domain experts are usually not familiar with software tools, to let them contribute to the design of the final application, the meta-design team develops proper *application templates*. The template is intended as a schema or a skeleton, which facilitates the assembling of some basic components. In the SSW approach, an application template aims at guiding the design activities of the stakeholders involved, who will behave as unwitting programmers, creating or modifying software artifacts without explicitly programming.

Another model proposed to support meta-design is Seeding, Evolutionary and Reseeding (SER) [41]. Instead of building a complete system at design time, the system design starts from seeds which are developed by meta-designers collaborating with end users; a subsequent evolutionary growth follows, and then a reseeding phase occurs. The seeding phase concerns the definition of the initial state (seed) of a software artifact, which will be used by end users to perform their activities. The reseeding of a software artifact is performed by any designer to modify the initial state of a software artifact. The evolving system continually alternates between periods of unplanned evolutions by end users and periods of deliberate restructuring and enhancement, involving end users in collaboration with designers. Compared with the SER model, the SSW model more explicitly considers that end users can even take on the role of meta-designers; it also blurs the distinction between design time and use time. Indeed, as shown in some examples in the next section, meta-design is not only performed by software engineers, but some end users themselves want to shape software artifacts that are used by other end users to design other artifacts. Moreover, the SSW model indicates how to support the designers in the reseeding phase, since there is ongoing communication among the SSWs of end users, professional developers and other stakeholders.

5. EUD in different application domains

This section presents some examples showing that EUD activities are needed in many contexts and illustrating how the SSW meta-design approach is carried out in the different application domains. In Section 5.1, the design of the electronic patient record in the medical domain is discussed. Section 5.2 presents a Web portal whose aim is to allow end users to contribute to the design of virtual shop windows for advertising purposes. Section 5.3 describes the EUD-based design of educational games in the cultural heritage domain. In Section 5.4, a recent project on product customization whose aim is to empower customers to create their own furniture is illustrated.

5.1. Designing the electronic patient record

Management of the Electronic Patient Record (EPR) is a key problem in the medical domain, which has been addressed by various researchers [42–44]. No generally accepted implementation of the EPR yet exists, because it is still commonplace that individual hospitals, and even specific departments within the same hospital, create their own procedures. Physicians, nurses and other operators in the medical field are reluctant to accept a common unified format; they want to customize and adapt the patient record to their specific needs, as various authors also observed [45,46]. Thus, the EPR is a natural target for EUD.

The patient record is a many-sided document: it is read and understood by very different people, not only physicians and nurses, but also the patients themselves, their relatives, etc., thus it must have the ability to speak different "voices" to convey different meanings according to people using it [46]. Patient records are official artifacts that practitioners write to preserve the memory or knowledge of facts and events that occurred in the hospital ward [47]. The patient record has two main roles: a short-term role to collect and memorize data keeping trace of the medical care provided during the patient's hospital stay; a long-term role in storing the patient's data for research or statistical purposes [48]. Accordingly, the specialized literature distinguishes between the primary and secondary purposes. Primary purposes include the demands for autonomy and support of practitioners involved in the direct and daily care of patients, while secondary purposes are the main focus of the hospital management, pursued in order to rationalize care provision and enable clinical research [46].

In most systems proposed so far for EPR management, pre-defined document templates and masks are usually imposed on practitioners, without considering the specific needs and habits of those who are actually using the EPR. The combination of requirements for both standardization and customization is a further element pushing toward the adoption of EUD techniques for managing EPRs. We present here our proposal to create an EPR whose structure and functionalities support the specific needs of each stakeholder involved, by allowing them to contribute to design and/or tailoring the EPR.

We conducted a field study at the "Giovanni XXIII" Children Hospital of Bari, whose purposes were: (1) to understand which kind of documents, tools and languages were used in order to identify the requirements of a system implementing the EPR; (2) to identify the stakeholders to be included in the design team. Unobtrusive observations in the wards, informal discussions, individual interviews were performed. The analysts involved in the study periodically observed different people during their daily work in the hospital. An important point emerged: in each ward, even in the same hospital, there are specific patient records; this is because different data need to be stored, depending on the specific ward. For example, in a children's neurological ward, information about newborn feeding must also be available, while in an adult neurological ward, information about alcohol and/or drug consumption is required. The patient records are actually composed of modules, each one containing specific fields for collecting patient data. Various hospital employees are interested only to a subset of such modules, and use them to accomplish different tasks, i.e. the nurse records the patient's measurements, the reception staff records the patient's personal data, the physician

examines the record to formulate a diagnosis, and so on. Moreover, the following main stakeholders who are involved in the EPR management were identified: (1) *practice manager*; (2) *head physicians*; (3) *physicians*; (4) *nurses*; and (5) *administrative staff*. In particular, the head physician has the right and the responsibility of the EPR to be adopted by physicians and nurses of his ward.

According to the SSW model, we created the meta-design team composed by software engineers, HCI experts and the practice manager, a domain-expert whose knowledge is necessary to design the EPR modules. The meta-design team created the SSWs for the different stakeholders, as well as the data modules, which are the basic component of the EPR, and the application template to allow each head physician to design the EPR for her/his ward by directly manipulating data modules in her/his SSW.

We briefly describe here the prototype system for the management of the EPR. The focus of this prototype is on the activities of the head physicians for shaping the EPR. The system is composed of a network of software environments (SSWs), each devoted to a different type of stakeholder to let them accomplish their tasks in a comfortable and suitable way. The SSWs used by physicians and the ones used by nurses of a specific ward result from the design activity performed by the head physician.

Let us consider a specific example. The SSW developed for the head physician of a specific ward allows him (in our case he is a man) to design the EPR tailored to the ward needs by choosing from the pre-designed modules those appropriate to the ward, and assembling them in the preferred layout. Fig. 1 shows the SSW for the neurology head physician. The working area of the SSW is divided in two parts: in the left part are the modules he can insert in the EPR (“Moduli Inseribili” in Italian), the right part shows the current design of the EPR (“Cartella Clinica”) performed by the head physician. For example, Fig. 1 shows, on the left, a module about Standard Growth Charts (“Misure Antropometriche all’ingresso”), which includes data about Weight (“Peso”), Height (“Altezza”), Head Circumference (“Circonferenza cranica”). Another module is about Feeding (“Allattamento”) and includes data about breast or artificial feeding; in cases of artificial feeding, more data about the type of milk and other aspects are included. The neurology head physician creates his tailored EPR by dragging and dropping the modules from the left part of his SSW to the desired position in the right part. In the example shown in Fig. 1, the neurology head physician has already inserted some modules, namely a heading reporting information about the hospital and the ward; the patient’s personal data, i.e. last name (“Cognome”), first name (“Nome”), birth date (“Data Nascita”), etc.; a module with Basic Laboratory Tests (“Routine Ematica”); a module for information about Required Consultations (“Consulenze Inviate”). In the figure, the head physician is dragging the module about Tests Performed Outside the Hospital (“Esami Fuori Sede”) to insert it in his EPR. Once the EPR design is completed, the head physician clicks on the Save button (“Salva Layout” in Italian). In this way, he has actually created a software artifact that will be used by the neurology ward personnel.

Fig. 2 shows how the EPR designed by the head physician appears in the SSW of the nurses of his ward (for privacy reasons, in figures dummy data are shown). A nurse primarily uses the EPR to input patients’ data. This end user does not have the EUD possibilities allowed to the head physician in his SSW: nurse’s tailoring is limited to modifying the layout of the EPR modules. For example, if her/his current activity is to insert patients’

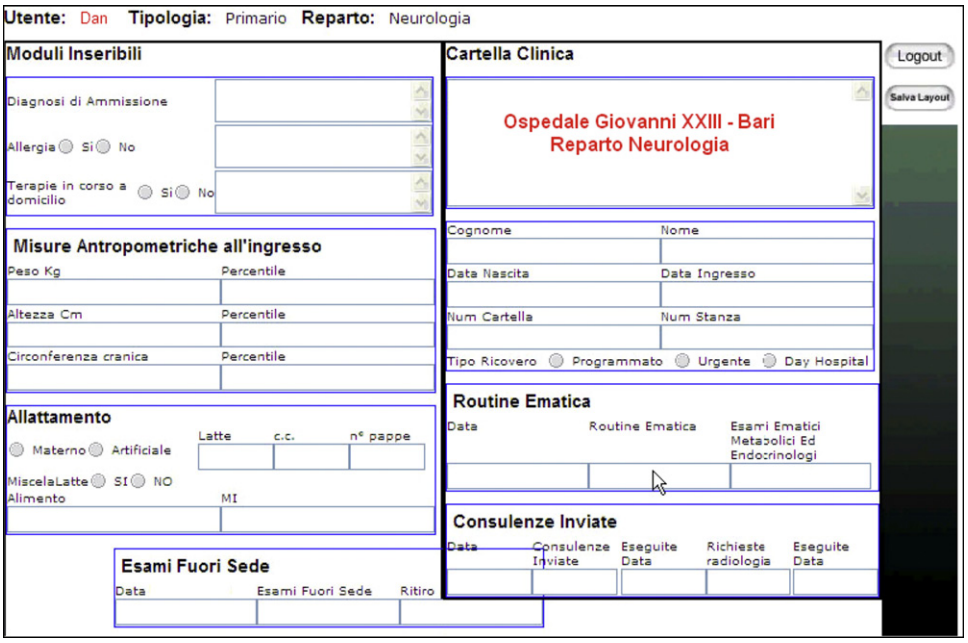

Fig. 1. A screen shot of the SSW for the user “Dan”, the head physician (“Primario”) of the Neurology (“Neurologia”) ward.

Fig. 2. A screen shot of the SSW for the user "Cic", a nurse of the Neurology ("Neurologia") ward.

data about "Routine Ematica", s/he can move this module to the top of the SSW by clicking on the "Up" button ("Su" in Italian).

In a similar way, the head physician designs the SSW to be used by the physicians of his ward. Over time, if necessary, the head physician can update the EPR for his ward by inserting new modules among those already designed. If what he requires does not yet exist, he refers to the meta-design team, who has to create new modules and make them available in the SSW of the stakeholders.

We are well aware that there is a long way from this prototype to a completely working system, but the formative user testing we performed with physicians and nurses confirmed our idea that users appreciate the possibility to shape the EPR according to their needs. A main goal of our work is to inform designers of EPRs of the value of EUD approaches in the medical domain.

5.2. Designing virtual showrooms on the Web

This section describes a recent project about the development of a Web portal for a company that provides advertisement spaces for shops of various natures through virtual windows appearing in the portal. The virtual windows are designed according to different patterns, which include various type of multimedia elements, such as text, pictures, Flash™ animations, videos, etc. The company sells virtual windows to shops for advertising purposes. The virtual windows are sold at different prices according to the complexity of the pattern (in terms of combined multimedia elements). An innovative feature of the portal is that the shops' owners are allowed to create and manage the content of their own virtual windows.

A preliminary study indicated the following main stakeholders of this system [49]: (1) *Web surfers*, who are people browsing the virtual windows; (2) *shop owners*, who provide multimedia contents to be shown in their virtual windows; (3) *editorial staff members*, who are company employees that manage the available patterns of virtual windows; and (4) *system administrator*, a company employee with some Computer Science knowledge. The system administrator is a member of the meta-design team that, working with software engineers and HCI experts, develops the SSWs for the different stakeholders and creates the patterns of virtual windows to be sold.

As an example of how a shop owner (a woman) uses her SSW to manage and craft her virtual windows, let us consider what happens when she wants a feature modified in her virtual window. Fig. 3 refers to the screen shot of the SSW that, in the central part, shows the virtual window whose pattern is composed of a textual description (left side) and a photo gallery (right side) with one large picture and three small pictures at the bottom. As illustrated in Fig. 3, the shop owner uses the tool available in her SSW (through the item "Richiesta modifica" in the left menu) to request the changes she wants on a widget of the interface she indicates with a mouse click. In this example the photo gallery has been indicated: a popup window is shown, where the user writes her request, namely to shows four pictures in the gallery.

The request is received by a member of the editorial staff and visualized in her/his SSW as an item in a table at the screen bottom (Fig. 4). The table shows all requests

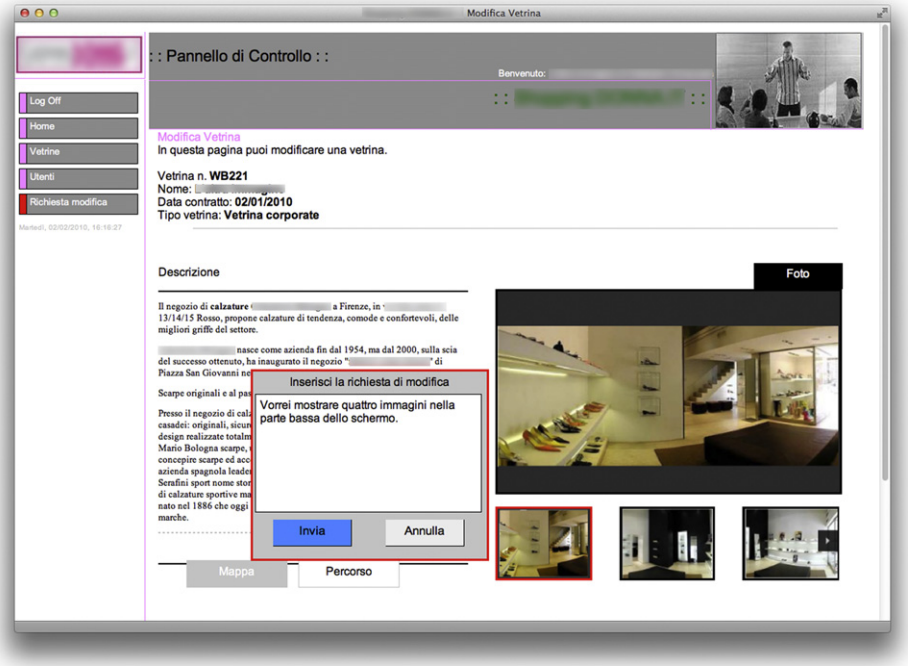

Fig. 3. Through the annotation tool in her SSW, the shop owner makes a request to the editorial staff.

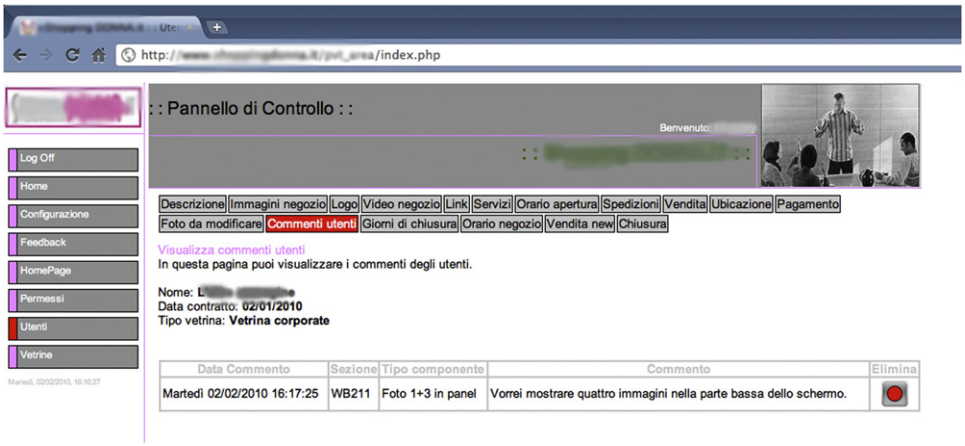

Fig. 4. A screen shot of the SSW of the editorial staff, showing that the request sent by the shop owner is visualized as an item in a table (at the bottom of the figure).

possibly issued by other shop owners. It is worth noticing that the representation of a same message is different in the two SSWs: in the SSW of the shop owner, the request is a text in an annotation; in the SSW of the editorial staff, it is an item in a table, with several attributes that, even if represented by codes, are understandable by the SSW users.

If the editorial staff member can directly manage the request, s/he performs the necessary software modification and communicates it to the shop owner. For example, if a pattern with more pictures exists, the editorial staff member makes it available in the SSW of the shop owner, who will update the content of the new pattern. Otherwise, the editorial staff member communicates in a similar way with the administrator in order to ask for the creation of the new pattern satisfying the shop owner's request [49].

5.3. *Designing educational games in the cultural heritage domain*

Over the years, we have been involved in research projects that are aimed at developing interactive applications for supporting visits to cultural heritage sites. In particular, pupils aged 10–13 years old have been addressed, designing educational games on newfangled

devices, such as cell phones and large multitouch displays; which can potentially arouse pupils' curiosity and to engage them in their learning activities. The excursion-game has been developed; it is a pervasive game on cell phones to be played by groups of pupils exploring outdoor cultural heritage sites, like archaeological parks [50,51].

The successful design of this type of applications requires the joint effort of several stakeholders with different/specific skills, namely: (a) *Education experts*, who contribute to specifying and reviewing requirements in design and evaluation of educational applications; (b) *Cultural Heritage* (*CH*) *experts,* who play a fundamental role in designing and developing applications that support visits to CH sites; (c) *visitors*, who use the developed applications. We describe here how the SSW model has been applied to create an environment (CH expert workshop), which allows CH experts to be co-designers; having no expertise in application design, they need a proper environment to carry out EUD activities. According to the SSW model, software engineers, together with HCI experts and other domain experts (e.g. education experts, CH experts), design software environments (SSWs) to be used by the communities of stakeholders. This meta-design activity also creates templates of the applications that support visitors of CH sites, as well as building blocks through which experts give content and functionalities to such applications.

Applications for visiting CH sites can be of different types, depending on the target visitors: while an excursion-game is suitable for schoolchildren, other types of guide are proposed to adults or more expert visitors (e.g. history experts or scholars). The CH expert workshop illustrated here evolve the one in [52]; it is inspired by YahooPipes [53] and allows CH experts to contribute to the design of the final application. It offers a visual design environment, application templates, and building blocks [54]. In other words, CH experts create the final applications from the templates by composing building blocks that allow them to shape user interfaces, functionalities and multimedia content without the direct help of professional developers. For instance, if the application is an excursion-game, the CH expert has to specify all the elements required by the game, namely the character to be impersonated and the prologue (i.e. the game introduction), missions to be performed, hints, places to be discovered (goals), 3D reconstructions of places, etc. [50]. In the example shown in Fig. 5, the CH expert is interacting with his own workshop for creating an excursion-game for the archeological park of Egnathia, in Southern Italy. First, he has selected *Excursion-Game* as application template. Thus, the workshop shows a screen in which he properly combines building blocks, chosen from the elements listed in the left toolbar. The screen shot in Fig. 5 shows a situation in which the expert has already defined the game prologue by connecting the *Prologue* building block to the root *Excursion-Game*. It has also defined some missions. Specifically, the building block *Mission* represents a single mission of the game. If the CH expert wants to add a mission, he drags the *Mission* building block from the toolbar to the main area of his workshop. Then he draws a line between the connection point of the *Mission* and the *Prologue* building block. If a building block allows 1:N connections, a *Connector* element is used. In Fig. 5, three missions, i.e. "Mission1", "Mission2" and "Mission3", have been connected to the Prologue through a Connector. The building block referring to "Mission1" is maximized because the CH expert is defining the text of the mission. For each mission, the CH expert is also defining goal and hints. For example, the goal of Mission1 is to reach the kiln ("Fornace" in Italian). Thus, when the mission is solved, the 3D reconstruction of the kiln is shown. The *Oracle Hint* building block permits CH expert to specify some hints for solving a mission.

The developed applications can be further modified by CH experts over the time. For instance, new missions can

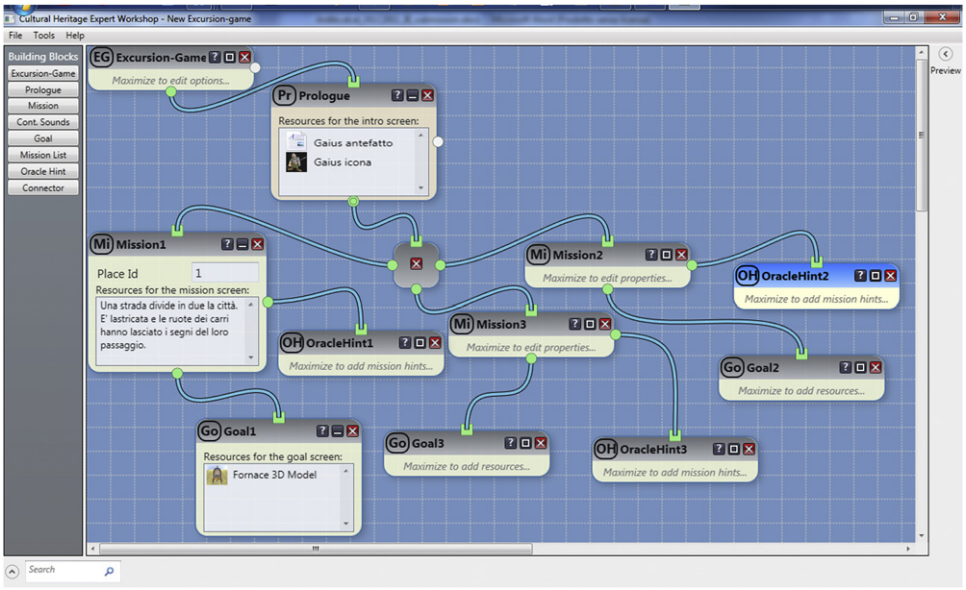

Fig. 5. A screen shot of the CH expert workshop.

be added to an excursion-game. Only when CH experts need new functionalities or new building blocks in their workshop, do they refer to software engineers who, collaborating with the other experts at the meta-design level, will update the CH expert workshop by providing the required elements.

5.4. Designing classic style furniture

The project described in this section is different from the previous ones, since it addresses design activities that people perform on goods they are buying. Thanks to the new technologies, the demand for narrow-target goods and services not available in traditional bricks and mortar stores can be as economically attractive as mainstream fare. More and more companies are taking into account opportunities to provide individual customers with unique personalized goods. Product configurators, also known as mass customization toolkits, design kits, or toolkits for user innovation and design, are now available [55–57]. They allow customers to change several aspects of the product, e.g. color, material, writings, etc. An example is the product configurator of IKEA^{TM}: the customer can select a product from the catalog, e.g. a table, and change some of its features, like the type of wood, the size, the color. A limitation of such configurators is that the changes allowed are constrained within a limited range of possibilities. For example, the IKEA^{TM} configurator does not permit customers to modify the drawers of a table as they like, because the personalization possibilities are pre-defined.

The project presented here has been motivated by Maiellaro s.r.l., a company in the Puglia region that produces classic style furniture. Since this type of furniture is very expensive, the company produces only pieces that are ordered by customers: they look at the company catalogs and provide a description of the piece of furniture they want, which may be composed of parts chosen from different items in the catalogs; they also specify dimensions, type of wood and other characteristics. Currently, the customer sketches his/her design on papers, which are sent via fax to the company for approval; the price of the new piece of furniture is also negotiated via fax or e-mail between company and customer. Aim of the project is to create a web-based system that allows customers to design the desired furniture and that manages the order process. Compared to the product configurators on the market, customers have much more freedom in designing their furniture; this is made possible by an ontology that models the possible composition of different parts in a whole piece. The ontology describes the components of each piece of furniture and their properties, i.e. it specifies colors, size, decorations, shapes, materials, etc., and provides all rules and constraints for assembling various components, so that the user is guided in the design of the desired pieces of furniture [58].

After a field study at the Maiellaro company for requirements analysis, we identified the following stakeholders: (1) the *managing director*, who supervises the company business processes and, in particular, is in charge of the approval of the order of new pieces of furniture, designed by the customers, which will then be

Fig. 6. A screen shot of the workshop for the company's customers.

inserted in the company catalogs; (2) *sales department employees*, who manage customers' orders in collaboration with the technical department; (3) *technical department employees*, who manage technical aspects of new pieces of furniture designed by customers and have the responsibility of updating the ontology when new catalogs or new furniture are added; and (4) *customers*, who order pieces of furniture they have created.

Following the SSW model, the system supporting the negotiation process between customers and company is composed by the workshops for each community of stakeholders, created by the meta-design team. The workshop for customers, which allows them to create the desired piece of furniture, is shown as an example in Fig. 6. The customer is browsing the catalog, shown at the top of the screen, where products are organized by category, e.g. Dressers (''Cassettiere''), Consoles (''Consolle''), etc. Fig. 6 shows that the user has selected the Console category thus in the central part of the screen the different consoles available in the catalog are displayed. The customer chooses a console of interest by clicking on its picture, and a thumbnail of that item appears in the box at the bottom of the screen. In Fig. 6 three consoles have been selected. When the customer has selected all items of interest, s/he goes on to create the piece s/he wishes. This piece can be either a specific item s/he found in the catalog, for which the customer only wants to modify certain features, e.g. type of wood, size, etc., or it can be the result of a more sophisticated design process, i.e. the composition of parts taken from different items. For example, the customer might desire a console made up of components taken from the selected items in the box at the bottom. S/He will go on with her/his customization process by clicking on the link ''Personalizzazione'' at the bottom right of the screen (see Fig. 6). A new screen will appear, where s/he can indicate the component of interest in each of the consoles previously selected. The reader can refer to [58] for more details. Once the user has completed the desired console, the sales department checks the received design by collaborating with the technical department. Once the new design is accepted and sales department and customer have agreed on the price (by communicating through their respective SSWs), the official order is delivered and the production of the console starts.

6. Conclusions

This paper has explored the roles of end users in the life cycle of interactive systems, determined by end users' increasing desire to become information producers, to shape the software they use and to contribute to the design of their own products. These roles are driving the trend towards a design paradigm that considers software design as an evolutive and never-ending process, which can be modeled as a design-develop-use-evolve cycle. This poses several challenges to professional developers, who have to become meta-designers, i.e. they have to design systems that permit end users to become designers themselves and to collaborate in their EUD activities with other stakeholders.

When performing EUD, end users behave as unwitting programmers, i.e. they have to be enabled to create or modify software, but this has to be permitted through software environments that recreate situations they are familiar with in their daily practices, so that they do this as part of activities they are highly motivated to perform, without being aware of programming. Such environments should comply with the ''gentle slope of complexity'' principle, thus they have to be open systems that can easily co-evolve with end users.

Another basic principle of our research work, addressed in this paper, is the diversity of end users [8,27]. This makes it necessary to: (a) perform a careful analysis of end users targeted by the system; (b) provide different support to different end users. As we have pointed out, this was Piero Mussio's major concern since the beginning of his work in 1970s.

To comply with these challenges, the model of meta-design we have illustrated in this paper prescribes that the different communities of end users as well as the other communities of stakeholders must be provided with different software environments, each suited to the specific skills and needs of the community it is intended for. The interactive system is thus designed as a network of software environments with proper communication channels among them [8]. By interacting with such environments, end users will perform their tasks (use phase), but some of them will also perform EUD activities, contributing to design, development and evolution of the system. This paper presents several examples of application of the design model we have developed. We hope it will provide insights for involving end users to contribute to the design of software artifacts and products they use.

Our work is in line with the so-called ''culture of participation'', which means that ''people are provided with the means to participate and to contribute actively in personally meaningful problems'' [59] (see also [60]). Contexts in which the culture of participation has been explored include architectural design and urban planning, design of computational artifacts, models of teaching and learning. The research we have carried out with Piero Mussio in the last decade has concentrated on models and approaches to system development that foster the culture of participation in the creation of computational artifacts.

Acknowledgments

This work was supported by the Italian MIUR through grant ''CHAT'', by the EU and the Regione Puglia through grant ''DIPIS'' and grant ''Tecnologie End-User Development per la personalizzazione di mobili classici italiani'', POR Puglia 2007–2013.

References

[1] G. Fischer, End user development and meta-design: foundations for cultures of participation, Journal of Organizational and End User Computing 22 (2010) 52–82.
[2] N. Zang, M.B. Rosson, Playing with information: how end users think about and integrate dynamic data, in: Proceedings of:

Symposium on Visual Languages and Human-Centric Computing (VL/HCC), IEEE Computer Society, Corvallis, Oregon, USA, 2009, pp. 85–92.
[3] C. Soh, S.S. Kien, J. Tay-Yap, Enterprise resource planning: cultural fits and misfits: is ERP a universal solution? Communications of the ACM 43 (2000) 47–51
[4] A. Molla, I. Loukis, Success and Failure of ERP Technology Transfer: A Framework for Analysing Congruence of Host and System Cultures, Working Paper Series, 2005.
[5] D. Fogli, L. Parasiliti Provenza, Information System Customization—Toward Participatory Design and Development of the Interaction Process, in: Proceedings of: International Conference on Enterprise Information Systems (ICEIS), Milan, Italy, 2009, pp. 72–77.
[6] C. Dörner, S. Draxler, V. Pipek, V. Wulf, End Users at the Bazaar: Designing Next-Generation Enterprise Resource Planning Systems, IEEE Software, IEEE Computer Society, Los Alamitos, CA, USA, 2009, pp. 45–51.
[7] Y. Dittrich, S. Vaucouleur, S. Giff, ERP Customization as Software Engineering: Knowledge Sharing and Cooperation, IEEE Software, Los Alamitos, CA, USA, 2009, pp. 41–47.
[8] M.F. Costabile, D. Fogli, P. Mussio, A. Piccinno, Visual interactive systems for end-user development: a model-based design methodology, IEEE Transactions of the Systems, Man, and Cybernetics A 37 (2007) 1029–1046.
[9] A.I. Mørch, G. Stevens, M. Won, M. Klann, Y. Dittrich, V. Wulf, Component-based technologies for end-user development, Communications of the ACM 47 (2004) 59–62.
[10] V. Wulf, V. Pipek, M. Won, Component-based tailorability: enabling highly flexible software applications, International Journal of Human–Computer Studies 66 (2008) 1–22.
[11] H. Lieberman, F. Paternò, V. Wulf (Eds.), End User Development, 9, Springer, Dordrecht, The Netherlands, 2006.
[12] G. Fischer, E. Giaccardi, Y. Ye, A. Sutcliffe, N. Mehandjiev, Meta-design: a manifesto for end-user development, Communications of the ACM 47 (2004) 33–37.
[13] M.F. Costabile, D. Fogli, G. Fresta, P. Mussio, A. Piccinno, Building environments for End-User Development and Tailoring, in: Proceedings of IEEE Symposium on Human Centric Computing Languages and Environments, IEEE Computer Society, Auckland, New Zealand, 2003, pp. 31–38.
[14] M.F. Costabile, D. Fogli, P. Mussio, A. Piccinno, End-user development: the software shaping workshop approach, in: H. Lieberman, F. Paternò, V. Wulf (Eds.), End User Development, 9, Springer, Dordrecht, The Netherlands, 2006, pp. 183–205.
[15] W.E. Mackay, Triggers and barriers to customizing software, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching through Technology, ACM, New Orleans, Louisiana, United States, 1991, pp. 153–160.
[16] B. Nardi, A Small Matter of Programming: Perspectives on End User Computing, The MIT Press, Cambridge, MA, 1993.
[17] A. Sutcliffe, N. Mehandjiev, Introduction special issue: end-user development, Communications of the ACM 47 (2004) 31–32.
[18] M. Burnett, C. Cook, G. Rothermel, End-user software engineering, Communications of the ACM 47 (2004) 53–58.
[19] A.J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M.B. Rosson, G. Rothermel, M. Shaw, S. Wiedenbeck, The state of the art in end-user software engineering, ACM Computing Surveys 43 (2011) 1–44.
[20] M.F. Costabile, D. Fogli, C. Letondal, P. Mussio, A. Piccinno, Domain-expert users and their needs of software development, in: Proceedings of the 2nd International Conference on Universal Access in Human–Computer Interaction, Lawrence Erlbaum Associates, Inc., Crete, Greece, 2003, pp. 532–536.
[21] A. Mørch, Three levels of end-user tailoring: customization, integration, and extension, in: M. Kyng, L. Mathiassen (Eds.), Computers and Design in Context, MIT Press, 1997, pp. 51–76.
[22] Y. Ye, G. Fischer, Designing for participation in socio-technical software systems, in: C. Stephanidis (Ed.), Universal Access in Human Computer Interaction. Coping with Diversity, Lecture Notes in Computer Science, vol. 4554, Springer, Berlin, 2007, pp. 312–321.
[23] C. Scaffidi, M. Shaw, B. Myers, Estimating the numbers of end users and end user programmers, in: IEEE Symposium on Visual Languages and Human-Centric Computing, IEEE Computer Society, 2005, pp. 207–214.
[24] M.F. Costabile, P. Mussio, L. Parasiliti Provenza, A. Piccinno, Advanced visual systems supporting unwitting EUD, in: Proceedings of the International Conference on Advanced Visual Interfaces (AVI), ACM, Naple, Italy, 2008, pp. 313–316.
[25] M. Petre, A.F. Blackwell, Children as unwitting end-user programmers, in: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), IEEE Computer Society, Coeur d'Alène, Idaho, USA, 2007, pp. 239–242.
[26] G. Stevens, V. Pipek, V. Wulf, Appropriation infrastructure: mediating appropriation and production work, Journal of Organizational and End User Computing (JOEUC) 22 (2010) 58–81.
[27] A. MacLean, K. Carter, L. Lövstrand, T. Moran, User-tailorable systems: pressing the issues with buttons, in: SIGCHI Conference on Human Factors in Computing Systems: Empowering People, ACM, Seattle, WA, United States, 1990, pp. 175–182.
[28] P. Mussio, R. Rabagliati, Analysis of water remote sensed data: requirements for data bases and data bases interactions, in: A. Blaser (Ed.), Data Base Techniques for Pictorial Applications, 81, Springer, Berlin, Heidelberg, 1980, pp. 369–411.
[29] P. Mussio, M. Pietrogrande, M. Protti, Simulation of hepatological models: a study in visual interactive exploration of scientific problems, Journal of Visual Languages and Computing 2 (1991) 75–95.
[30] P. Mussio, M. Finadri, P. Gentini, F. Colombo, A bootstrap approach to visual user-interface design and development, The Visual Computer 8 (1992) 75–93.
[31] International Organization for Standardization, ISO 13407: Human-Centered Design Process for Interactive Systems, 1999.
[32] D. Schuler, A. Namioka, Participatory Design: Principles and Practices, Lawrence Erlbaum Associates, Inc., 1993.
[33] Google, iGoogle, 2011, 〈http://www.google.com/ig〉, accessed on April 28, 2011.
[34] V. Pipek, V. Wulf, Infrastructuring: toward an integrated perspective on the design and use of information technology, Journal of the Association for Information Systems 10 (2009).
[35] K. Beck, C. Andres, Extreme Programming Explained: Embrace Change, 2nd edition, Addison-Wesley Professional, 2004.
[36] T. O'Reilly, What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software 2005, 〈http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html〉, accessed on April 28, 2011.
[37] Software Engineering Institute, Ultra-Large-Scale Systems: The Software Challenge of the Future, Carnegie Mellon University, Pittsburgh PA, 2006, 〈http://www.sei.cmu.edu〉, accessed on February 22, 2011.
[38] M.F. Costabile, P. Mussio, L. Parasiliti Provenza, A. Piccinno, Supporting end users to be co-designers of their tools, in: V. Pipek, M.B. Rosson, B. de Ruyter, V. Wulf (Eds.), End-User Development, Lecture Notes in Computer Science, vol. 5435, Springer, Berlin, Heidelberg, Germany, 2009, pp. 70–85.
[39] K.E. Iverson, Notation as a tool of thought, Communications of the ACM 23 (1980) 444–465.
[40] M. Spahn, C. Doerner, V. Wulf, End user development: approaches towards a flexible software design, in: Proceedings of the 16th European Conference on Information Systems, Galway, Ireland, 2008.
[41] G. Fischer, Seeding, evolutionary growth and reseeding: constructing, capturing and evolving knowledge in domain-oriented design environments, Automated Software Engineering 5 (1998) 447–464.
[42] G. Hardstone, M. Hartswood, R. Procter, R. Slack, A. Voss, G. Rees, Supporting informality: team working and integrated care records, in: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, ACM, Chicago, IL, USA, 2004, pp. 142–151.
[43] M. Berg, E. Goorman, The contextual nature of medical information, International Journal of Medical Informatics 56 (1999) 51–60.
[44] B.R. Winthereik, S. Vikkelsö, ICT and Integrated care: some dilemmas of standardising inter-organisational communication, Computer Supported Cooperative Work 14 (2005) 43–67.
[45] C. Morrison, A. Blackwell, Observing end-user customisation of electronic patient records, in: V. Pipek, M.B. Rosson, B. de Ruyter, V. Wulf (Eds.), End-User Development, Lecture Notes in Computer Science, vol. 5435, Springer, Berlin, Heidelberg, 2009, pp. 275–284.
[46] F. Cabitza, C. Simone, LWOAD: a specification language to enable the end-user develoment of coordinative functionalities, in: V. Pipek, M.B. Rosson, B. de Ruyter, V. Wulf (Eds.), End-User Development, Lecture Notes in Computer Science, vol. 5435, Springer, Berlin, Heidelberg, 2009, pp. 146–165.
[47] M. Berg, Accumulating and coordinating: occasions for information technologies in medical work, Computer Supported Cooperative Work 8 (1999) 373–401.
[48] G. Fitzpatrick, Integrated care and the working record, Health Informatics Journal 10 (2004) 291–302.

[49] C. Ardito, B.R. Barricelli, P. Buono, M.F. Costabile, A. Piccinno, S. Valtolina, L. Zhu, Visual mediation mechanisms for collaborative design and development, in: C. Stephanidis (Ed.), Universal Access in HCI, Part I, HCII 2011, Lecture Notes in Computer Science, vol. 6765, Springer, Heidelberg, 2011, pp. 3–11.
[50] C. Ardito, P. Buono, M.F. Costabile, R. Lanzilotti, T. Pederson, A. Piccinno, Experiencing the past through the senses: an M-learning game at archaeological parks, IEEE Multimedia 15 (2008) 76–81.
[51] M.F. Costabile, A. De Angeli, R. Lanzilotti, C. Ardito, P. Buono, T. Pederson, Explore! possibilities and challenges of mobile learning, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI), ACM, Florence, Italy, 2008, pp. 145–154.
[52] C. Ardito, R. Lanzilotti, An EUD approach to the design of educational games, International Journal of Distance Education Technologies (IJDET) 9 (2011) 25–40.
[53] Yahoo! Inc., YahooPipes, 2011, 〈http://pipes.yahoo.com/pipes/〉, accessed on July 28, 2011.
[54] C. Ardito, P. Buono, M.F. Costabile, Involving end users to create software supporting visits to cultural heritage sites, in: Proceedings of the 9th ACM SIGCHI Italian Chapter International Conference on Computer–Human Interaction: Facing Complexity, ACM, Alghero, Italy, 2011, pp. 157–162.
[55] N. Franke, M. Schreier, U. Kaiser, The "I designed it myself" effect in mass customization, Management Science 56 (2009) 125–140.
[56] A. Trentin, E. Perin, C. Forza, Overcoming the customization-responsiveness squeeze by using product configurators: beyond anecdotal evidence, Computers in Industry 62 (2011) 260–268.
[57] T.W. Simpson, Product platform design and customization: status and promise, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 18 (2004) 3–20.
[58] C. Ardito, B.R. Barricelli, P. Buono, M.F. Costabile, R. Lanzilotti, A. Piccinno, S. Valtolina, An ontology-based approach to product customization, in: M.F. Costabile, Y. Dittrich, G. Fischer, A. Piccinno (Eds.), End-User Develpment, Lecture Notes in Computer Science, vol. 6654, Springer, Berlin, Heidelberg, 2011, pp. 92–106.
[59] G. Fischer, Understanding, fostering, and supporting cultures of participation, Interactions 18 (2011) 42–53.
[60] H. Jenkins, Confronting the Challenges of Participatory Culture: Media Education for the 21st Century, MIT Press, Cambridge, MA, USA, 2009.