Pre-print version
Ardito C., Buono P., Costabile M.F., Lanzilotti R., Piccinno A., Zhu L., 2014. On the transferability of a meta-design model supporting end-user development. *Universal Access in the Information Society*, 14 (2), 169-186, issn: 1615-5289, doi 10.1007/s10209-013-0339-7.

On the Transferability of a Meta-Design Model Supporting End-User Development

Carmelo Ardito[1], Paolo Buono[1], Maria Francesca Costabile[1], Rosa Lanzilotti[1], Antonio Piccinno[1], Li Zhu[2]

[1]Università di Bari Aldo Moro, Dipartimento di Informatica, via Orabona 4, 70125 Bari, Italy
{carmelo.ardito, paolo.buono, maria.costabile, rosa.lanzilotti, antonio.piccinno}@uniba.it
[2]Earlymorning, Via dei Valtorta 5, 20127 Milano, Italy, julie@earlymorning.com

Abstract: *Purpose*. The availability of different methods (and models) that support the design and evaluation of interactive systems raises a question about the transferability of such methods between application sectors and domains. The transferability refers to the selection and application of a method in a development context, qualifying it for the interactive system in hand. The transferability process should help to identify the main features of the new contexts of use, also taking into account that the system to be developed has to ensure universal access. Moreover, it should allow designers to capitalize on previous development experiences in a systematic way.

Method: In order to analyze some of the many challenges determined by the transfer process, this paper reports experiences of transferring a meta-design model, whose aim is to support the design of systems that enable people to perform end-user development activities. The model is further developed when applied in another application domain.

Result: A meta-design model can be used in a novel context supporting the design of systems for users performing development activities. Based on the reported experiences on active people involvement, hints for the transferability of any model are provided.

Conclusion. People can actively contribute to system design, development and evolution over time using the novel approach.

Keywords: End-User Development, Meta-Design, Design Model, Environments for Shaping Software

1. INTRODUCTION

Transferability of design and evaluation methods is a challenging research issue. Because of the proliferation of interactive systems in an increasing number of application domains, implemented on various devices, it becomes fundamental to consider methods that have already proved to be effective and efficient, learn from their application experiences and try to use them for the new design problem to be managed. Despite the interest in method transferability, there is very

little in the literature about this. The ongoing COST Action IC0904 "TwinTide" is stimulating researchers on this topic, and some are starting to provide their contributions. An example is the initial framework, proposed in [1], to address factors that might favor transferability. Other researchers are trying to propose more specific guidelines to support the transfer process. It would be desirable that such guidelines could help to identify the relationships among the richness of the different contexts of use and available technologies and allow designers to capitalize on previous development experiences in a systematic way. Thus, method transfer is crucial in software organizations, especially when they are constrained by strict economic conditions.

An important requirement of interactive systems is that they should be usable by all the intended types of users, in all the intended locations, at any intended time and through any intended media and/or device. This is in accordance with the so-called *Universal Access*, i.e. the systematic effort to apply principles, methods and tools of universal design, in order to develop technologies accessible and usable by all citizens, independently of their cognitive and/or physical abilities. Method transfer should therefore guarantee the transfer of concepts, techniques and artifacts, devoting particular attention to ensure the usability of the final applications for all the intended users and contexts of use.

This paper contributes to the debate on model and method transferability by describing how a meta-design model, which supports the design of systems that enable people to carry out activities of End-User Development (EUD), has been applied to different application domains [2-5]. Enabling EUD entails providing end users, who in most cases are not technologically skilled, with appropriate environments and tools that allow them to contribute to the design, development and evolution over time of software artifacts. The purpose of this paper goes beyond presenting the model, which has been described in previous publications, and focuses on how it can be transferred to different domains. The authors reason on the transferability process and realized that it can be facilitated by identifying the key elements of the model, which have to be specified, in order to successfully apply the model in a new domain. Thus, the model has been analyzed, its key elements have been clearly defined and they are presented in this paper for the first time. Moreover, the specific steps for the application of the model are described. Two examples of several experiences of applying the model to create interactive systems in different domains are revisited, focusing on the steps to be performed and on the implications of transferring the model. Finally, it is shown why, in a particular situation, the model has evolved, in order to address the specific requirements of the new contexts of use, where collaboration of different stakeholders is very much stressed.

The paper is organized as follows. Section 2 illustrates the meta-design model: its motivation, its value with respect to related approaches, its key elements and its graphical representation. Section 3 describes the process to apply the model in different contexts and Section 4 reports two case studies in which the model has been used. Section 5 illustrates another case study, which led to an evolution of the model, and presents the new model. Section 6 reports some related work. Section 7 concludes the paper by summarizing the main reasons for the successful application of the model, some of which can be easily generalized to become useful hints for transferring any model.

2. THE SOFTWARE SHAPING WORKSHOP MODEL

The first ideas about the model, on which the design approach is based, appeared in [6] and [7]. Later, the approach was refined, as shown in successive papers [2], [8], [9]. The novelty of the model presented in this section is the description of its key elements, on which designers have to focus when transferring the model to a different context. The last part of the

section illustrates the model graphical representation.

2.1 Motivation

Exponential technological advances push end users to evolve from a traditional role as passive information consumers to a more active one. Users are increasingly willing to shape the software they use to adapt it to their needs, tasks and habits, by manipulating and tailoring software artifacts, in order to create new configurations or new designs. End-User Development and End-User Software Engineering are terms coined to refer to these end users' activities [7], [10], [11]. Tasks that are traditionally performed by professional software developers are thus transferred to end users, who become co-designers of the tools and products they will use. Of course, end users have to be supported in these new roles as 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 the end users.

Over the years, work has been carried out on the creation of software infrastructures that may support EUD activities, as well as knowledge creation and sharing among the stakeholders who should be involved in the design team. Indeed, all stakeholders of an interactive system, including end users, are 'owners' of a part of the problem: software engineers know the technology, Human-Computer Interaction (HCI) experts know human factors, graphic designers know how to create an appealing graphical design, end users know the application domain, etc. They should all contribute to system design by bringing their own expertise. However, such stakeholders are very different types of people. In accordance to universal usability principles, they need software environments, languages and tools adequate to their cultural background and skills, through which they can contribute to shape software artifacts [2]. Moreover, because both users and systems evolve during time, all stakeholders should take part in a continuous evolution of a system [12].

The design of systems that permit EUD activities requires a shift in the design paradigm, which must move from user-centered and participatory design to meta-design, characterized by two main phases [2], [13]. The first phase consists of creating the design environments that allow system stakeholders to participate in the design (meta-design phase). The second phase consists of the design of the final applications, carried out by the joint work of the various stakeholders, who collaborate through their design environments (design phase).

According to the *meta-design* paradigm, all system stakeholders, including end users, are active members of the design team. The professional developers involved in the traditional design are the team of meta-designers, who create *software environments* through which the other stakeholders, acting as designers, can be creative and can adapt the software to fit their specific needs. They can create and modify elements (objects, functions, user interface widgets, etc.) of the system of interest, and exchange the results of their activities to converge to a common design. Such software environments have been called *Software Shaping Workshops* (SSWs) since they are seen as virtual laboratories (workshops) through which users of different types contribute to shape software artifacts [2], [6], [7]. The model capturing this meta-design approach is therefore called SSW model.

2.2 Relationships with other meta-design approaches

Fischer et al. introduced meta-design as a conceptual framework to foster EUD [13]. Recently, Fischer referred to the SSW model as one of the most significant meta-design approaches [14]. He also cited the Hive-Mind Space (HMS) model, which is an evolution of the SSW model and is described in Section 5. A third model is SER (Seeding, Evolutionary and Reseeding) [15]. The name comes from the fact that, instead of building a complete system at

design time, system design starts from seeds, which are developed by meta-designers in a participatory team involving end users. A subsequent evolutionary growth follows, and then a reseeding phase occurs. The seeding phase concerns the definition of the initial prototype, which will be used by end users to perform their activities. The reseeding is performed by any designer to modify the initial state of a software artifact, on the basis of the evolutions caused by end users. 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 allows end users more design power and the possibility to even take on the role as meta-designers. It also blurs the distinction between design time and use time. 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.

Koehne et al. focused on the specific domain of virtual worlds (VWs), in order to assess how meta-design can inform the design of virtual worlds [16]. Their analysis showed that indeed meta-design can provide designers with useful tools for involving end users in the design of VWs, such as massive multiplayer online role-playing games, like 'Lord of the Rings Online', and open-ended VWs, like 'Second Life'.

Maceli and Atwood analyzed the history of the design and use of interactive systems by taking into account the role of end users. Both in literature and practice, it emerged that end users, often adopting a trial-and-error strategy, were always trying to modify the tools they use. Thus, in accordance with the view presented in this paper, these authors confirm that meta-design successfully supports this trend, because it aims at enabling end users to become co-designers of the systems, even when they are using them, thus intertwining design time and use time [17].

2.3 Key elements of the SSW model

While reasoning on the transferability process of a model, the authors realized that this process can be facilitated by identifying the key elements that have to be specified for the successful application of the model. For the SSW model, the following key elements were identified: 1) the *stakeholders* involved in the meta-design process; 2) the *SSWs* to be developed for each community of stakeholders; 3) the *interaction languages* used in the SSWs, customized to the specific experts to which the SSW is devoted; 4) the *building blocks* available in the SSWs, i.e. the basic elements that have to be composed to create the final applications; 5) the *application templates*, which guide the building block composition; and 6) the *communication channels* among SSWs, which allow information exchange among the different stakeholders. Such elements are described here in more details.

SSWs

Software Shaping Workshops (*SSWs*) are software environments provided to the various community of stakeholders, through which they either use the developed applications or act as designers, modifying or even creating software artifacts according to their specific needs.

The term *workshop* comes from the analogy with an artisan workshop, e.g. joiner's or smith's workshop, which is the workroom where the artisan finds the tools, specific for her/his job, to carry out her/his activities. Artisans organize their workroom and specialize the tools they use according to their own habits, skills and activities to be performed. For example, both joiner and smith use a hammer, but the hammer of the smith is different from the one of the joiner, since it is used to shape iron, which is a much harder material than wood. Following this analogy, the SSWs are tailored to needs, culture, and skills of the specific community of stakeholders they are devoted to. They provide all and only the tools that allow their users to perform the activities of interest, in a way that is suitable for

them.

The rationale for providing different SSWs is to ensure universal access and universal usability of interactive systems, given the diversity of users. The slogan "one size fits all" cannot be applied to interaction environments. Even if belonging to a same community, people could experience difficulties when they interact with a certain environment, due to their different background or skills; they have to be provided with software environments whose user interface is customized to them. Moreover, users should have the possibility to further tailor their environment. There might be different levels of tailoring, going from a simple setting of parameters (to specify some preferences) up to modifying/assembling components or eliminating unnecessary functions that often disorient them, or even creating new software artifacts.

Stakeholders

According to the meta-design paradigm, various stakeholders are actively involved in the design team. In all cases in which the model has been applied to develop systems with a visual interface, the design team always included professional developers, namely, software engineers, HCI experts and graphic designers, but also different communities of domain experts and end users.

– *Software engineers* have the technical responsibility of the project. Examples of their SSWs are Microsoft Visual Studio® or Eclipse, since they are professional developers able to use traditional design tools and programming languages. They are responsible of the meta-design team, which creates the SSWs for the other communities of stakeholders, through which all experts may collaborate to create the final application. Later in the software life cycle, i.e. after the software is delivered, if there is the need to evolve it, they implement new functionality to be added to the SSWs of the other stakeholders.

– *HCI experts* bring their knowledge on human factors to the design. Using her/his SSW, an HCI expert takes care of usability and user experience, carries out evaluations and gives feedback to the other members of the meta-design team.

– *Graphic designers* contribute with their expertise to the look and feel of the user interface. Their SSW includes their typical design tools.

– *Domain experts* are the owners of the domain knowledge and of the problem in hand [18]. As such, they have to be actively involved thorough the whole design-development-evolution process of an interactive system. In early phases of design, they contribute to specifying and reviewing requirements, and to the design of the SSW prototypes. Later, they use the tools available in their SSW to contribute to the design of the final application.

– *End users* are persons who use computer applications as part of daily life or daily work. Often, there are different communities of end users that need to collaborate to reach a common goal (see the examples in Section 4). It is worth noticing that even end users belonging to the same community have different skills and needs, depending on their levels of experience. For instances, in a hospital ward, physicians may have different expertise and perform different tasks. Thus, different types of end users have to be provided with an SSW suitable to them. As competent practitioners, end users perform their daily activities, determining solutions to problems in their domain and possibly adapting virtual tools to their needs. When adapting such tools, they are performing EUD activities, i.e. they are migrating from a mere user role to a design role.

Interaction languages

According to Iverson [19], a notation developed by people during years of experiences is a tool of thought. Indeed, each community of people

has developed a notation that properly expresses the concepts and activities of that community. For example, in the medical domain, physicians communicate among them using specific technical languages and annotate documents, e.g. radiographies, with specialized graphical notations. The language for the human-computer dialog adopted in each SSW has to be derived from the languages and notations used in the daily practices of the community the SSW is devoted to. It is evident that tools, languages and interaction paradigms for the software engineer SSW are different from those in a domain expert SSW. Software engineers use an integrated development environment (e.g. Microsoft Visual Studio® or Eclipse) for programming in C# or Java or other programming languages, and create different types of diagrams (e.g. deployment, object, sequence, use case diagrams) by means of CASE tools. Usually, domain experts do not know programming; they can contribute to the design if they are provided by appropriate means, e.g. if their SSW allows them to direct manipulate visual elements, which resemble objects and tools used in their daily practice.

Building blocks

Building blocks are basic elements available in a SSW, which can be conveniently composed for creating the final application. Building blocks available in the SSWs represent entities (e.g. data, tools) used by people in their real working environment; they provide specific contents and functions to operate on such contents. As will be shown in the example regarding the Electronic Patient Record described in Section 4, the building blocks correspond to specific sections of data identified in the traditional patient record, which is composed together by physicians and nurses according to their goals.

Application templates

An application template can be intended as a schema or a skeleton of the final application, which facilitates the assembling of the building blocks. The application template guides the design activities of those non-technologically skilled stakeholders, who can thus create or modify software artifacts without explicitly programming. Different application templates can be available in an SSW; the user selects one of them, depending on his/her goals. It will be shown in Section 4 that in the case of the Electronic Patient Record, the application template is a form that includes several sub-forms, which are the building blocks representing the different sections of patient's data.

Communication channels

In several development practices, communication and collaboration among stakeholders take place through channels that are separated from the actual software, e.g. phone, e-mail, thus limiting active participation in collaborative design. In order to allow different stakeholders, working from different SSWs, to contribute to shaping the final application, communication among the SSWs has to be guaranteed. For example, at design time HCI experts and domain experts might exchange messages about usability problems of the application under construction. HCI experts and domain experts also communicate with software engineers when new functionalities or tools have to be implemented in some SSWs. The SSWs with the updated tools are then delivered to the different stakeholders. At use time, when working in a team, end users exchange data related to their current task to achieve a common goal.

2.4 Graphical representation of the SSW model

As it has been argued in the previous section, the interactive system is created, and also modified and evolved at use time, with the collaboration of different stakeholders, who participate through their own SSW. They also exchange information among them. The graphical representation of the model is a hierarchical network, as shown in Figure 1, in

which SSWs are organized in three levels according to the meta-design paradigm.

- *Meta-design level* includes the SSWs for software engineers and for the other professional developers (HCI experts, graphic designers, etc.), who create and maintain all the SSWs in the network. The interaction languages in such SSWs, especially that one of software engineers, are characterized by high computational power (Turing Machine equivalent) but cannot in general be understood and managed by non-technical people, i.e. they have low usability with respect to them.
- *Design level* includes SSWs for domain experts participating to the design of the final system. Their SSWs provide interaction languages specialized to the users' culture, which have less computational power than the ones used by software engineers, since they permit a limited set of operations. However, they are more usable by non-technical people.
- *Use level* includes SSWs devoted to end users to perform their well-defined set of activities using domain-oriented languages that reflect their traditional notations. These SSWs are characterized by a low computational power, permitting a limited set of functions, those of interest for their end-users community. On the other hand, usability is high: words, icons, symbols shown in the SSW user interface have to be familiar and significant for end users, in order to be correctly interpreted and used.

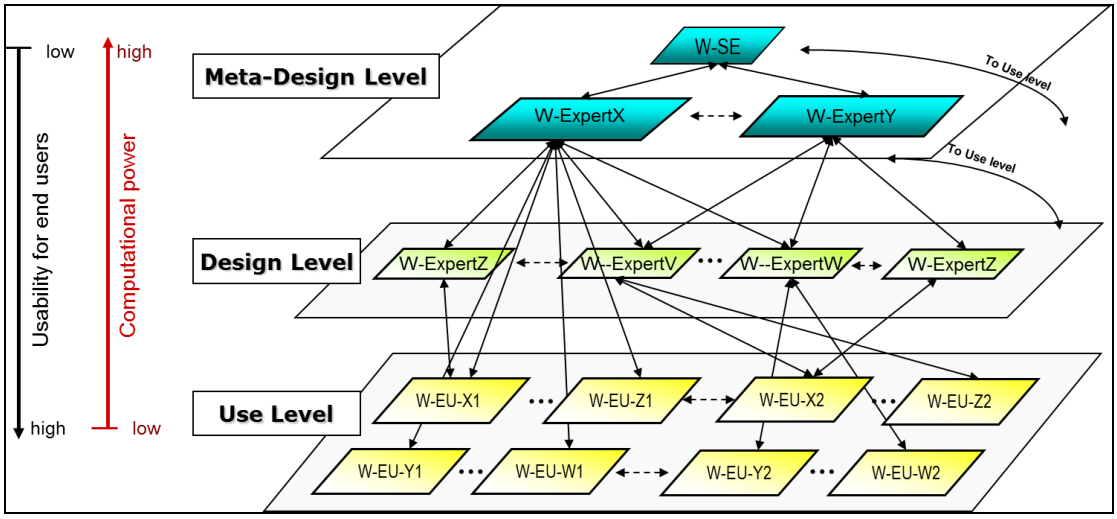

Figure 1. Graphical representation of the SSW model as a network of SSWs

3. THE PROCESS OF APPLYING THE SSW MODEL

In order to apply the SSW model to the design of a new interactive system, two main steps have to be considered: 1) a pre-phase, in which a contextual enquiry is carried out to study the domain, to identify and analyze the main system stakeholders, and to acquire the necessary knowledge to inform the model-based design; 2) a core-phase, in which the interactive system is designed and implemented as a network of SSWs.

The contextual enquiry consists of

observations, interviews, and informal discussions in the field [20]. The stakeholders related to the specific case have to be first defined, namely domain experts and end users; it is fundamental to understand their background, working procedures and habits, as well as documents, tools, and languages they use. A document with a detailed description of user and system requirements is the output of the pre-phase, which provides specific information for defining the key elements of the model. In other words, this pre-phase supports the identification of *who* are the stakeholders, *what* they need in their SSWs and *which* interaction languages have to be implemented, *which* building blocks and application templates have to be provided to facilitate non-technical people in contributing to the design, *which*, *when* and *how* information is exchanged among the different stakeholders.

The first activity of the core-phase is to create a meta-design team, which includes professional developers, i.e. software engineers, HCI experts and graphic designers, as well as representatives of domain experts and end users. They collaborate to the specification of the other key elements of the model: functionality of the SSWs, interaction languages, building blocks, application templates, and communication channels. Special care must be devoted to define the SSW interaction languages most appropriate for domain experts and end users, who often have very limited technical skills and want to interact with the system in the simplest possible way. Indeed, for the type of systems the authors have developed, it has been found that the most successful interaction paradigms for these types of users are form fill-in and drag-and-drop. Such paradigms capitalize on the availability of application templates, which represent software frameworks to be completed with specific building blocks.

The meta-design team carries out the actual design and development. Through their SSWs at the top level of the network, the professional developers create prototypes of the SSWs for different experts, which include the building blocks and their supporting templates. By interacting with their own SSWs, such experts, who know working context and habits of end users, design and develop the SSWs customized to the different end-user communities. End users use their SSWs to carry out their tasks. During time, end users' needs may evolve, so that they require to perform new tasks not supported by their specialized SSW. An end user may communicate these new needs to a domain expert who, by possibly collaborating with other experts, evolves the SSW accordingly. Whenever the domain expert cannot satisfy the end user's request, s/he refers to professional developers, who will develop new software artefacts and update the SSWs.

The core-phase refers to the design-development-evolution of the overall system based on the key elements of the model and organized as a network of SSWs. In other words, in the core-phase, application templates, building blocks, interaction languages are implemented and made available in the created SSWs, which are organized in a network and exchange all the necessary information through the communication channels.

4. TWO EXAMPLES OF APPLYING THE SSW MODEL

Several examples of applying the methodology derived from the SSW model to create systems in different domains are reported in previous papers [2], [3], [21], [22]. Two examples are revisited here to focus on the steps to be performed and on the implications of transferring the model-based methodology to another domain.

4.1 Designing the electronic patient record

The first case study refers to the medical domain. The authors collaborated with the physicians of the "Giovanni XXIII" Children Hospital of Bari, in Southern Italy, to develop some applications to support their work (see for example [23-25]). In some meetings, the advantages of an Electronic Patient Record (EPR) for managing data about patient history

were discussed. They clearly remarked the difficulties of accepting one of the many proposals of EPR, because they impose to practitioners predefined document templates and masks. Physicians, nurses and other operators in the medical field are reluctant to accept such unified templates; as various authors also observed [26-28], they want to customize and adapt the EPR to their specific needs. Thus, the EPR is a natural target for EUD and was considered as an interesting case study to apply the SSW model.

The pre-phase supplemented the discussions with the physicians with unobtrusive observations performed in the wards by HCI experts, namely some of the authors of this paper. They also individually interviewed the key persons of the hospital management, some ward head physicians, physicians and nurses. This allowed the researchers to understand the work-flow in relation to the patient record and to get familiar with documents, tools, languages and task performed. The following stakeholders for the EPR management were identified: 1) *practice manager*; 2) *head physicians*; 3) *physicians*; 4) *nurses*; 5) *administrative staff*, 6) *patients*. In particular, the head physician has the right and the responsibility to decide about the patient record adopted by physicians and nurses of his ward. The analysis of the work activities clearly showed that each ward personnel use their specific patient record; this was because different data have to be stored, depending on the ward. The paper patient record is composed of different pages (often structured forms), on which the needed data are reported. If some data are not necessary in a ward, the corresponding pages are empty. On the contrary, some specific forms may be added. In an adult neurological ward, for example, information about alcohol and/or drug assumption is needed, while it is not required in a children neurological ward, where information about milk nutrition is required for newborn patients. The analysts proposed that the different forms included in the paper patient record could be the building blocks to be provided in the SSWs for the domain experts.

As first step of the core-phase, the meta-design team was created, including software engineers, HCI experts, a graphic designer and the practice manager, who is the domain expert whose knowledge is necessary to design the EPR modules. The team analyzed the data to be included in the EPR and decided how to group them, in order to have small meaningful modules, e.g. personal data, blood pressure data, anthropometrical data, etc., which, in most cases, reproduced the forms of the paper patient record. Such modules constitute the building blocks of the EPR. The meta-design team also agreed that the most suitable application template for creating the EPR by composition could be a form in which the user will drag and drop the building blocks (sub-forms) representing the different sections of patient's data he wants to have in the EPR.

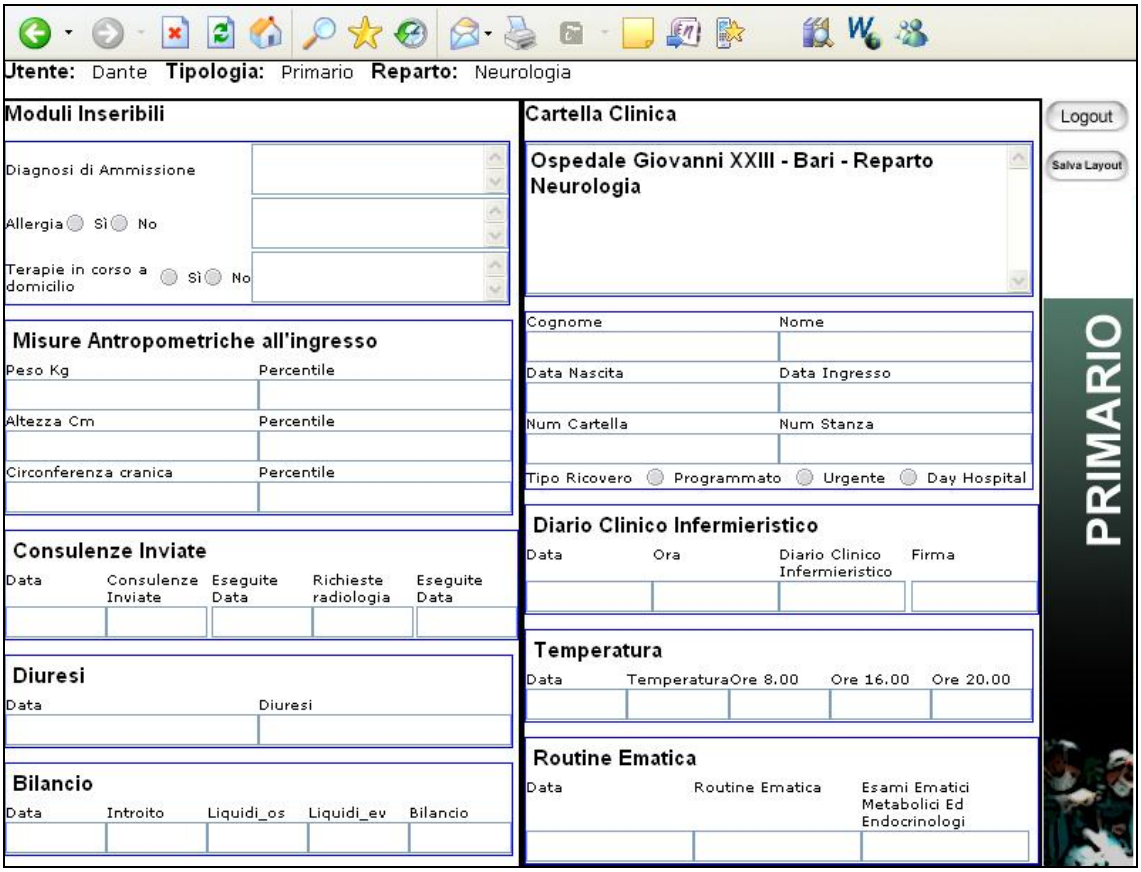

Figure 2. A screen shot of the SSW that the head physician of the Neurology ("Neurologia") ward uses for creating the EPR for the personnel in his ward by dragging the data modules from the left side to the right side.

A brief description of the SSW that has been developed for the neurology head physician, to allow him/her to design the EPR to be used in his/her ward, follows. The SSW is illustrated in Figure 2. On the left side of the SSW, all data modules (i.e. building blocks) that can be inserted in the EPRs used in the hospital ("Moduli Inseribili" in Italian) are shown, namely: Standard Growth Charts ("Misure Antropometriche all'ingresso"), External Advice ("Consulenze Inviate"), Diuresis ("Diuresi") and further modules not visible in the figure. The neurology head physician creates the EPR ("Cartella Clinica") tailored for his ward by dragging and dropping the modules from the left part to the desired position in the right part. Once he has finished the composition of the EPR (right list of modules in Figure 2), he saves it ("Salva Layout" button on the right of Figure 2); in this way, the neurology head physician creates the EPR that will be available in the SSW for the neurology ward personnel, which is actually a software artifact to be used for managing patient data in that ward.

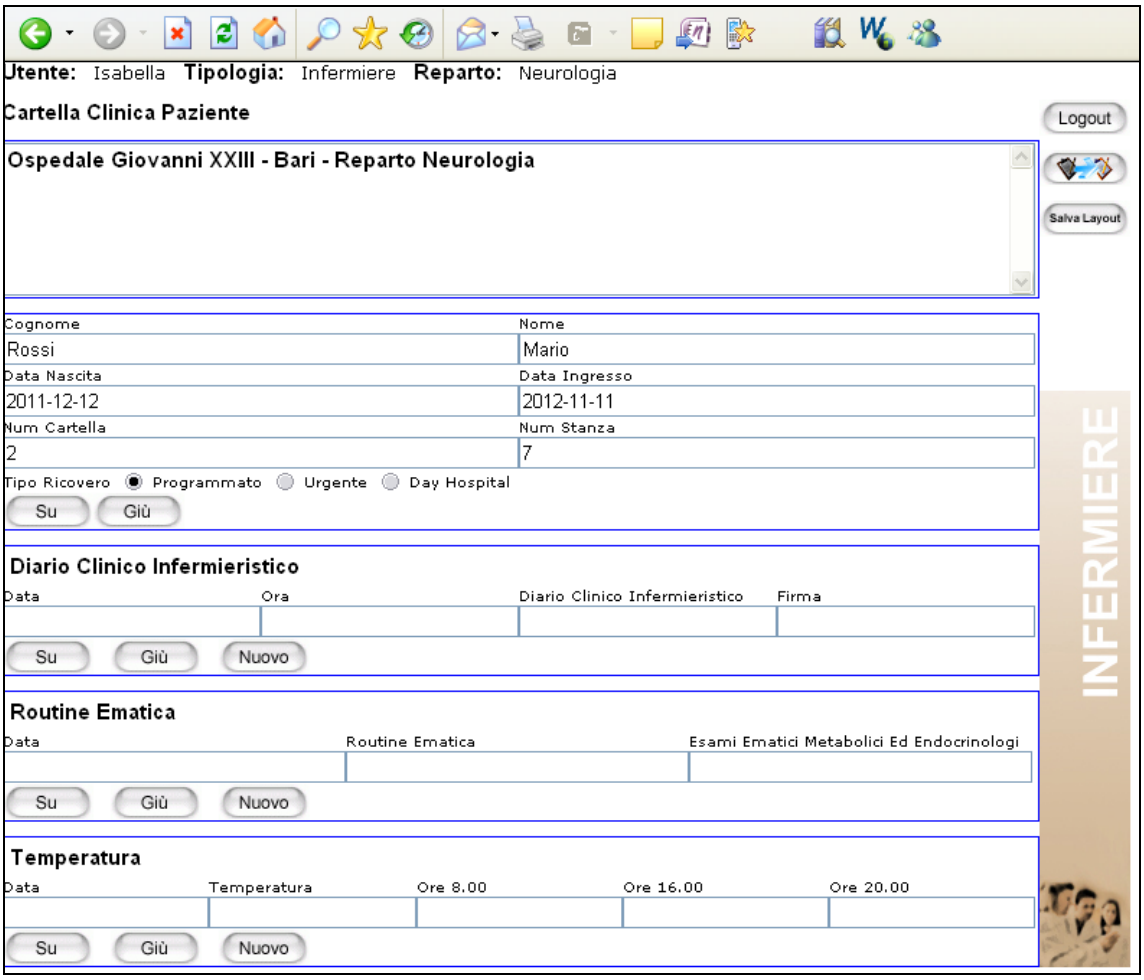

Figure 3. A screen shot of the SSW for the user "Isabella", a nurse of the Neurology ward.

Figure 3 illustrates how the EPR designed by the neurology head physician appears in the SSW devoted to the nurses of his ward (for privacy reasons, dummy data are in the figures). As shown in Figure 3, the EPR consists of only those modules made available by the head physician, i.e. Nurse's Clinical Diary ("Diario Clinico Infermieristico", where the nurse takes note of date and time when possible problems emerged and how they were fixed), Hematic Routine ("Routine Ematica", where the nurse reports when she took a blood sample for a specified test), Temperature ("Temperatura", where for a specified day, the nurse reports patient's temperature at 8 a.m., 4 p.m. and 8 p.m.); further modules, not visible in the figure, are available. 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' data about the clinical nurse diary, s/he can move the correspondent module "Routine Ematica" 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 created. 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.

4.2 Designing educational games in the cultural heritage domain

At the IVU Lab of the University of Bari, the authors have worked in the context of archaeological parks since several years, being involved in research projects aiming at developing interactive applications for supporting visits to cultural heritage sites. In particular, educational games have been developed to be played by school children with the support of smart phones [29] and large multi-touch displays [30], [31]. The discussions within the participatory design team that was set up and the user studies performed at archaeological parks during such design experiences highlighted that applications for visiting cultural heritage sites can be of different types, depending, in particular, on the target visitors: while an excursion-game is suitable for pupils, other types of guide are proposed to adults or more expert visitors (e.g. history experts or scholars). Also, the need to update the developed applications due to the frequently changes of the exhibitions emerged. In this context, the traditional design practices, in which only professional developers create and modify software, become very resource demanding (cost of developers, time to get new releases, etc.). Thus, it was decided to apply the SSW approach to create a system that could allow domain experts to actively participate in the creation and the update of applications supporting visits to a cultural heritage site.

As usual, the pre-phase of the process for applying the SSW model to this new context consisted of a contextual enquiry. The domain knowledge gained during the previous design experiences was complemented with the analysis of further requirements that emerged by observing how several professional guides were leading groups of tourists and school students of different age in some archaeological parks of the Apulia region, in Southern Italy. Interviews and focus groups involving guides, park staff, visitors were instrumental for capturing more details on the visit organization and on the whole visit experience.

The core-phase started by setting up the meta-design team. Beyond professional developers, the specific case required 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. The most peculiar SSW in this context is the one for CH experts, through which they can create the final applications for visitors, without the direct help of professional developers [32]. As stated, the aim was to develop different applications, addressing different types of visitors. Thus, special attention was devoted to the definition of the application templates, which represent the skeleton of the final application to be used by visitors. Three application templates are currently available in the CH expert SSW: 1) *excursion-game* for cell phones; 2) *puzzle game* for different devices (e.g. large multi-touch screens, cell phones); 3) *museum guide*, which is a more traditional guided tour provided on cell phones. The building blocks provide specific content and various user interface functions (e.g. showing content, inputting data, searching, zooming). 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 by the children playing the game, the prologue (i.e. the game introduction), the missions to be performed, hints, places to be discovered (goals), 3D reconstructions of places, etc. [29].

In the example shown in Figure 4, the CH expert is interacting with his SSW for creating an excursion-game for an archaeological park. First, he has selected *Excursion-Game* as application template. Thus, the SSW shows a screen in which he properly combines building blocks, chosen from the elements listed in the left toolbar. Each application template guides the CH expert composition by providing, in a way that is transparent to end user, the rules for composing

the building blocks in a meaningful way, according to the activities to be performed by the visitors in the site. The screen shot in Figure 4 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, i.e. activities that players have to perform.

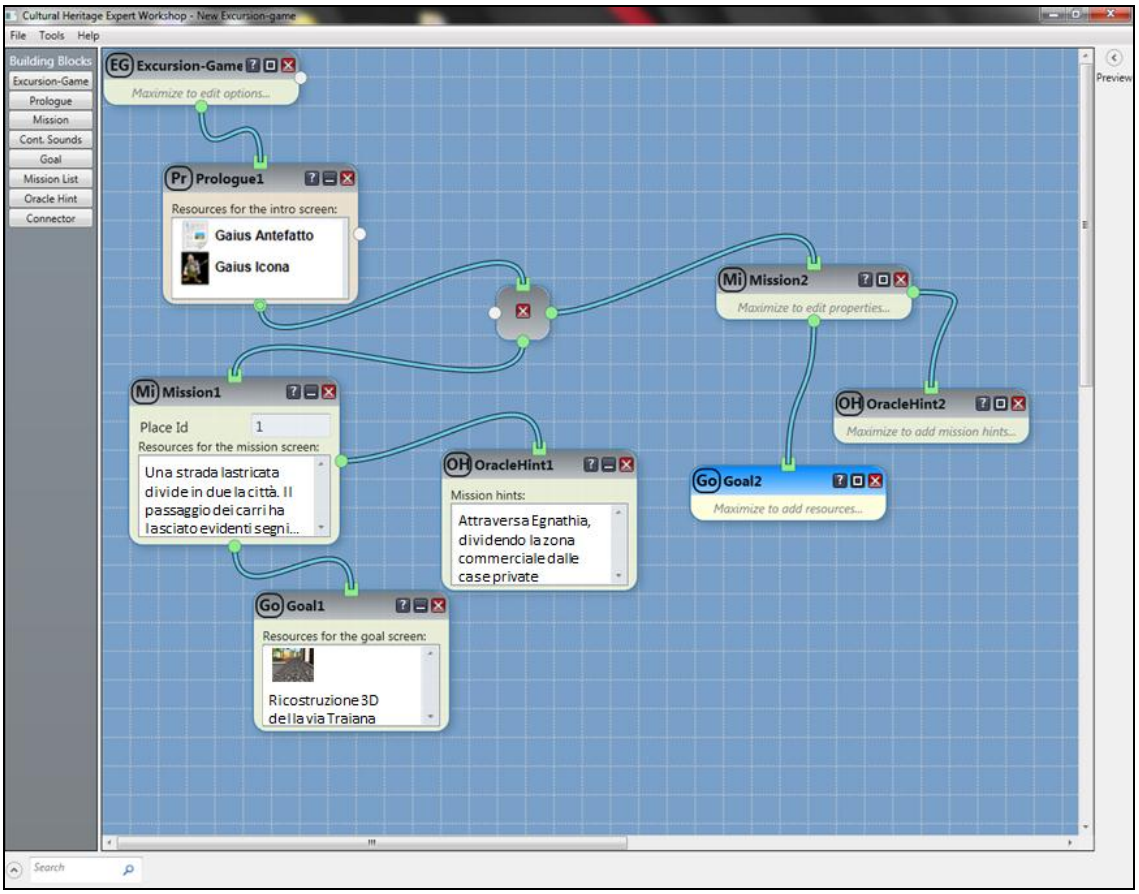

Figure 4. A screen shot of the CH expert SSW.

The developed applications can be further modified by CH experts over time. For instance, new missions can be added to an excursion-game. If CH experts need new functionalities or new building blocks in their workshop, they refer to professional developers at the meta-design level, who will update the CH expert workshop by providing the required elements.

4.3 Feedback from the case studies

The SSW model was applied to other contexts in different domains than the ones described in this paper (see, e.g. [9], [3]). Always, the feedback received from the involved end users was positive and encouraging. The domain experts appreciated very much the meta-design approach, which allowed them to contribute to the design of the final applications. The head physicians the authors worked with at the hospital were never satisfied of the various proposal of EPR they had examined, which forced the adoption of a format not adequate to the needs of their wards; thus, they liked a lot the opportunity to eventually shape the EPR tailored to their wards. The CH experts reported that it was much easier implementing new games or guides for visiting a cultural heritage site. Another positive remark of the domain

experts was that they felt to be actually aided in their designer role both by the appropriateness of the tools available in their design environment and by the possibility to ask the meta-design team for help or modifications.

Undoubtedly, applying the SSW model to new contexts presents some challenges. One is about the involvement of domain experts and end users in the meta-design team. This was especially critical in the case of the EPR, since it is well known that physicians are always in a hurry and are reluctant to devote part of their time to activities not directly related to their work. Eventually, some physicians enjoyed to collaborate, being keen on technology and excited by the potentiality of the new approach to EPR design. Moreover, when applying the meta-design model to a new domain, it is necessary to create or customize the SSWs for the different stakeholders, according to the requirements emerged in the pre-phase. Regarding the SSWs for the meta-design team, it is often enough to update the tools of the already available SSWs; for example, software engineers could need different plug-ins in Eclipse, according to a specific technology they want to use; or HCI experts could need a set of specialized heuristics for their evaluations. A bigger effort is necessary when considering the SSWs for domain experts and end users, since they depend very much on the specific context. However, the strategy adopted was to release a first seed, namely a beta version of such SSWs implementing the main functions, so that domain experts and end users have as early as possible the possibility to participate in shaping their software. Moreover, they can soon evaluate if the implemented tools comply with their needs and they are stimulated to possibly ask for new functions satisfying emerging requirements.

5. EVOLUTION OF THE SSW MODEL

This section describes how and why the SSW model has been evolved to be applicable in contexts where the main concerns are social interaction and personal creativity, and there are no specific domain experts participating in the design (as head physicians and CH experts in the two case studies considered in the previous section). More specifically, the aim is to support the collaboration of web developers and other stakeholders, including end users, in rapid prototyping of web applications. It is well known that web applications can be used by any type of users, so it is not possible to distinguish and characterize different communities of end users, who may perform very different tasks with the application, as it happens, for example, in the medical domain for the case of the EPR. Moreover, requirements like creativity, openness, and flexibility are emphasized on the Web. It is pointed out in [33] that much human creativity arises from activities that take place in a social context, i.e. in the interaction with other people and with artifacts that embody group knowledge and previous thinking. A promising approach to support social creativity might be arranging informal ways for end users to share experiences, in order to articulate their collective knowledge [33].

One of the objections raised regarding network representation of the SSW model with the three different levels according to different activities (meta-design, design, use), is that it appears too much abstract and rigid. The separation of the levels makes difficult to understand that some users can have different roles, e.g. being a designer or an end user of the final application. For example, in the EPR case study, a nurse's SSW is at some times at use level in the network and at other times at design level.

To address social creativity, it is necessary to enable end users moving fluidly from one role to another [34], supporting a graceful migration to higher levels of customization and design. Additionally, since the circumstances of designers' actions are continuously changing, the model should consider the possibility to accommodate the unforeseeable contingencies of situated actions [35]. During the design of a prototype, the decision making process is situational, creating

and testing on the spot. The temporal dimension is compressed by several connected time spans to moment-by-moment improvisations.

Therefore, cultivating and supporting creative and collaborative design of interactive software requires evolving the SSW model to emphasize that infrastructure/context for collaboration and for encouraging ad-hoc activities are required. The aim is to design open systems that can be easily extended and integrated with other tools, which in turn become creation resources. Given these considerations, the Hive-Mind Space (HMS) model evolves the SSW model, focusing on participatory creativity and thus stressing a lot the communication channels which allow people's collaboration [36].

5.1 The representation of the HMS model

The Hive-Mind Space (HMS) model aims at representing the meta-design approach to creative collaborative design. Like the SSW model, the HMS model supports three different levels of participation. However, it aims at making more explicit that design and use are intertwined rather than distinctive, and that the levels of participation are different modes of interaction that are possible in the same SSW. The graphical representation of the HMS model emphasizes that participants can act as meta-designers, designers or users, deciding moment-to-moment how to interact with artifacts. In other words, a user working in his/her environment (SSW) might use an artifact at a certain time (use level), tailor it at another time (design level), or even customize it to the needs of other people (meta-design level). In the HMS model, different stakeholders may operate at the same level, at the same time, and in a peer relationship, although they work on different issues. There is the potential for self-evolution of every community. The name Hive-Mind Space derives from the analogy with self-organized systems such as ant colonies. A hive mind implies a bottom-up system, which does not have a clearly defined hierarchy, and follows the organization rules that could lead to an emerging structure and collective intelligence greater than the sum of each individual.

The HMS model is represented in Figure 5. Different stakeholders are provided with their own environment, customized to their working habits. Their collaboration with peers, in order to modify or evolve the software, is stressed in the model; thus, technical means to relate and integrate users' and developers' views have to be provided, in order to permit a seamless way of moving between use and design of software. Such technical means include modeling languages, architectures supporting multilevel design and development, but also mediation agents. In a collaborative design context, *mediation agents* are defined to facilitate the communication among different stakeholders across the ecosystem, since they support them to reach a common understanding [37], [38]. Human actors playing a certain role in the collaboration exchange messages. Such actors may belong to different communities, thus the software environments they use are equipped with an engine that acts as a mediator by translating the sender's message into the interaction language that the receiver uses in her/his environment. Indeed, messages sent by a human actor consist of artifacts of the interactive system, called *boundary objects*, that are received and interpreted differently by another human actor according to her/his background and expertise [39]. It is worth remarking that face-to-face collaboration also considers boundary objects, i.e. blueprints, sketches, drawings, which are used during discussions within design teams.

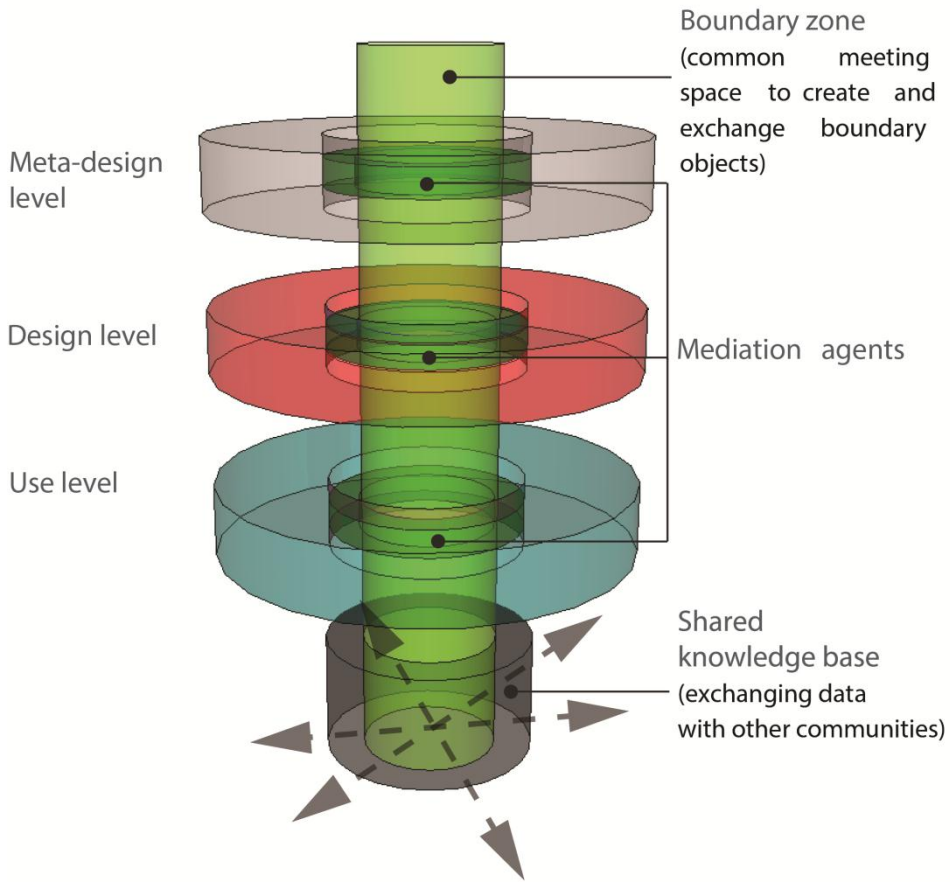

Figure 5. The Hive-Mind Space Model.

The inner cylinder in the HMS model (*boundary zone* in Figure 5) serves as a communication channel, through which collaborating designers exchange boundary objects. All other stakeholders can access and use these boundary objects, as well as jointly construct new boundary objects to be stored in the *shared knowledge base*. Boundary objects are utilized to mediate communication among the different stakeholders. As already described, mediation agents support this communication by allowing a same boundary object to be represented differently within individual environments. In order to highlight their function of mediators between the boundary zone and a specific level, in Figure 5 each mediation agent is represented by a ring around the inner cylinder.

5.2 Applying the HMS model

The HMS model has been applied for implementing MikiWiki, a web-based meta-design system aiming at supporting meta-design, social interaction, and design and communication among peer end users [40]. MikiWiki stands for 'meta-wiki'. Wikis are a collection of pages that can be edited by anyone, at any time and from anywhere. They are a popular format for sharing knowledge in both academia and industry domains [41]. The wiki architecture matches many conceptual aspects of the HMS model. MikiWiki leverages some features of a regular wiki, namely collaboration, rich context, openness and dynamic links. Beside normal wiki functionality, MikiWiki extends wikis with development functionality, so that the system can be modified and evolve along with collaboration practice. Collaborative and communication features in MikiWiki are not built-in; rather, they are made available as under-designed “nuggets” on top of the system.

A nugget is a page, which can be used as an embeddable component within another page, in order to create sharable remixable components.

MikiWiki has been used to rapidly prototype and try out new user interface designs. Mockups can be easily stored and shared among different stakeholders, and annotation tools and boards are used to provide feedback. As an example, reference is made to the design of prototypes of a web application, called Energy Feedback, for home energy management [40]. Consumers will access this application using an iPad tablet, as they want to monitor home energy consumption anytime and anywhere.

In the Energy Feedback case study, both web developers and other users are meta-designers able to take existing pieces of code and combine them in some way to create a new functionality [42]. They might not invent brand new solutions, classes or frameworks; rather, they tinker with existing components in creative ways to solve their task. There are also graphic designers with some web design experiences. They continuously adapt nuggets to create a design space where design tasks can be performed. The generic mockup environment designed through MikiWiki should allow end users to design the final energy feedback application, tailoring the visualization of their energy usage data so that it will be meaningful for them.

Figure 6 presents the three levels of participation in relation to the iPad mockup environment. In the initial phase of collaboration (meta-design level), meta-designers created different nuggets to design the mockup environment. They created application templates for iPhone mockup design, which could be used directly by designers. At the second phase (design level), designers adapted initial iPhone mockup environment, and created an iPad design environment for designers and users. In this case, the mockup design environment is composed of three nuggets, which are a toolbox, a canvas, and a trash icon shown in the middle of Figure 6. This creation process is iterative and possibly performed in collaboration with meta-designers. Designers also performed some meta-design activities opportunistically, when they realized an immediate advantage. For example, some of them were able to change the data visualization by accessing the program code and inserting some new tags by simply reproducing similar tags. Moreover, they analyzed and reasoned on the navigation and interactions of the final application. Some designers had difficulties in getting started, but once they understood the underlying logic, they started tinkering with sample code and reconfiguring nuggets and the environment to create different visualizations.

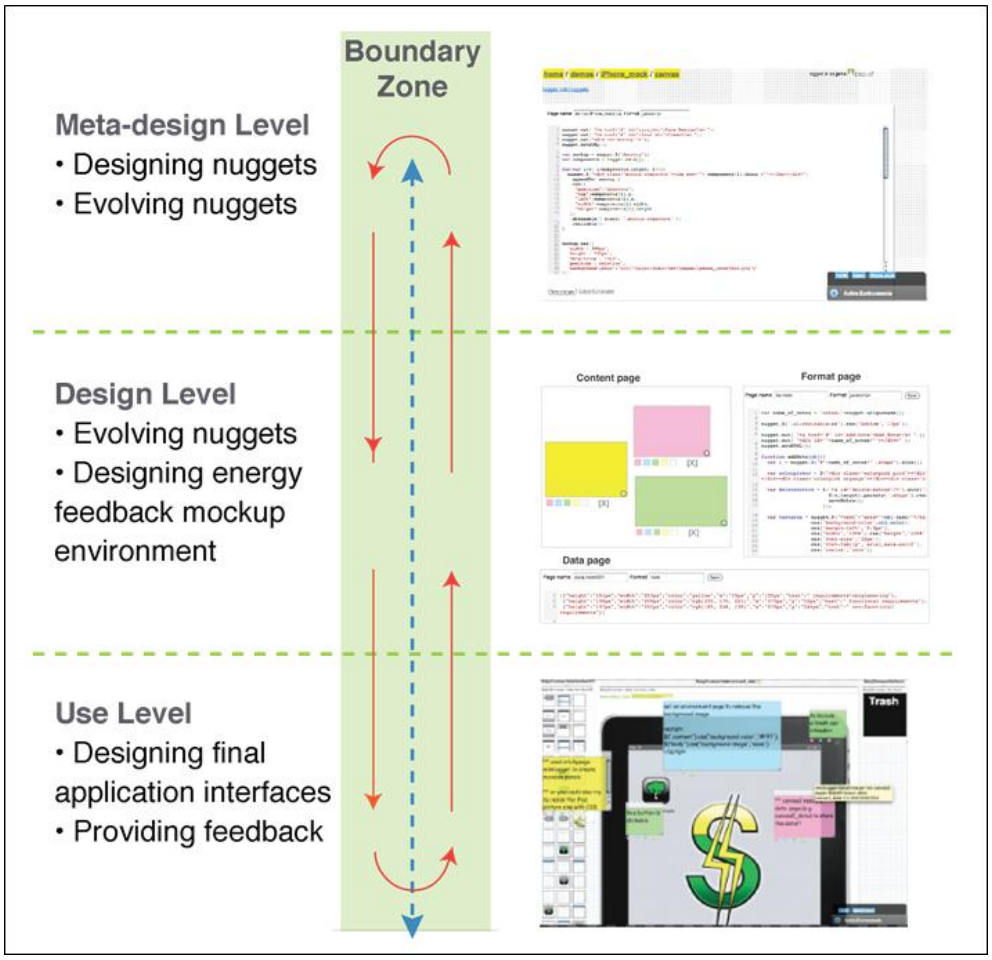

Figure 6. Three different levels of participation.

The bottom of Figure 6 refers to the use level. End users used the available canvas and tools (created at design level) to generate an iPad mockup by simply dragging and dropping components. They can also share their mockup, so that other end users can tailor it to their needs. Moreover, at the design level, the toolbox and the iPad canvas can easily be replaced by those for a new mobile device for another design case. At the use level, users can rate, annotate and discuss the mockups by using the MikiWiki markup language. Users can also create notes to ask meta-designers for new features and modifications or to annotate the application interfaces with suggestions and comments.

All design activities took place in the same environment. The levels of participation were not predefined, but emergent in terms of users' skills and roles, the latter being highly dynamic. The HMS model stresses that design and use levels are intertwined and all participants switch among different modes of participation. Another characterizing aspect is that the HMS model suggests breaking down the initial system into smaller seeds, which can be inspected, adapted and continuously evolved, thus not only blurring the distinction between design time and use time, but also easing the distinction between the evolutionary growth and the reseeding phases, which are tightly coupled together within the same system. A system based on the HMS model (e.g. Energy Feedback) can be seen as a collection of seeds (initial prototypes of components), namely under-designed software artifacts that combine fixed and clearly understandable parts with incomplete

representations, which support the integration of various end users' perspectives. They foster end users to draft solutions and leave enough room for proposing creative add-ons during design and use.

6. RELATED WORKS

The SSW model described in this paper aims at supporting a meta-design approach to create interactive systems that enable people to perform EUD activities, thus becoming co-designers of the software tools they use. In presenting the model in Section 2, a comparison has been made with other meta-design approaches. The SSW model was defined on the basis of various experiences carried out in previous years about the developments of several systems that should support professional people, such as physicians, geologists, mechanical engineers, in performing their daily work (see for example [3], [7]). Several works were influential for this research. The most significant ones are summarized in the following section.

The SSW model suggests a participatory approach, in that the design is performed by an interdisciplinary team, including domain experts and end users, or their representatives [43]. Participatory approaches exploit techniques derived from social science that support communication and collaboration within the interdisciplinary team. In some participatory design approaches, like cooperative prototyping [44], prototyping is a cooperative activity of users and designers. Prototypes are developed by software engineers, then discussed and modified with users, and possibly experienced by them in work-like situations. However, in this approach prototypes just represent an interactive digital evolution of paper-based mockups: real systems are then re-programmed and all modifications require a large programming effort. The SSW model suggests the asynchronous communication of ideas through the exchange of annotated prototypes. Prototyping becomes a cooperative incremental activity, in which all stakeholders may participate through their SSWs, operating according to their own view. Domain experts and end users contribute to the creation of software tools that they can tailor and, to some extent, even program themselves, as in *participatory programming* [45]. The so-called "translation problem" among different stakeholders is overcome by the SSWs model because it allows each stakeholder to participate in the design process through an SSW customized to their needs; mediation agents support the translation of a message from the interaction language of the sender to the one of the receiver.

The work performed within EUD-Net, a network of Excellence on End-User Development, funded by the European Commission during 2002-2003, provided several hints for the SSW model. The meta-design paradigm was also very much discussed within EUD-Net [8], [13]. End-user development entails the active participation of end users in the software development process, in order to modify, and even create, software artifacts. EUD activities may go from simply setting parameters, to integrating existing components, up to extending the system by developing new components. Mørch proposes three types of tailoring by end users: customisation, integration, and extension [46]. *Customisation* usually consists of configuring a set of preferences performed by the user through a preference form, by setting parameters for the various configuration options the application supports. *Integration* goes beyond customization and allows users to add new functionalities to an application by linking together predefined components, without accessing the underlying implementation code. *Extension* refers to the case in which the application does not provide, by itself or by its components, any functionality that accomplishes a specific user need, thus adding a new functionality generates a radical change in the software. A more specific classification of EUD activities is reported in [8]. The current approach to EUD presented in this paper takes into account all these types of tailoring. It also builds on the results in [47] and [48], which

report empirical studies showing that, since the nineties, end users were willing to perform EUD activities. In particular, Nardi [48] conducted empirical studies on users of spreadsheets and CAD software, and found that those users were able to perform activities that generate new software artifacts when they had a real motivation for doing so.

Software technology has advanced to the point that it is possible to build tools enabling end-users to design systems by direct manipulation of various widgets. Several researches capitalize on this and describe technologies for component-based design environments (e.g. [49]), libraries of patterns, and templates. Giving end users ways to easily create their own programs is important, but it is not enough, since end users need support for quality aspects of the software life cycle. For example, referring to the case of errors in end-users' programs, such as formula errors in spreadsheets, Burnett et al. discuss a strategy that gives end users the ability to perform quality control methods [50]. Some researchers indicate End-User Software Engineering as a discipline that aims at providing new kinds of technologies to support end users to improve the quality of the software artifacts they develop [11].

The SSW model emphasizes Universal Access by addressing systems that offer different software environments, each specific for a community of stakeholders, since they have different expertise and roles. A similar concern is identified in DAISY (Design Aid for Intelligent Support System), a design methodology for building decision support systems in complex, experience-centered domains [51]. DAISY provides a technique for identifying the specialized needs of end users within a specific range of domain experience. DIGBE (Dynamic Interaction Generation for Building Environments) is a system that creates end-user interfaces adapted to the multiple end users with different roles that collaborate to the management of a building control system [52].

An interesting research field refers to model-based approaches for the development of user interface to access information and services on the Web through several kinds of devices, e.g. laptops, palmtops and cellular phones [53]. Some adopt declarative models [54] to be used by model-based user interface design environments to build the final user interface. An approach used by some researchers is to specify an abstract representation of a user interface, i.e. a representation not linked to a specific interaction modality (visual, aural, tactile, etc.) neither to a specific deployment. The user interface is then adapted, in a completely or partially automated way, to the multiple devices and/or contexts of use (e.g. [18], [55]). Unlike the SSW model, that prescribes different software environments each devoted to a user community, such approaches support the development of multi-device user interfaces but do not specifically address the diversity of their end-user communities.

The overall design methodology based on SSW model remarks the importance of both context of activity and working organization and, as suggested by Activity Theory [56], considers software systems as artifacts having a mediating role between objects and subjects of activities. In fact, end users work in a context outside the computer and are required to apply their knowledge to reflect on the current situation and decide what to do next. Following Schön [57], it is assumed that end users perform their activity as competent practitioners. The interactive system should support end users in exploiting their competence and skill.

7. DISCUSSION AND CONCLUSION

The proliferation of interactive systems in many application domains forces software companies to pay great attention not only to the reuse of software, but also to the reuse and easy adaptation of any tool that could support their design and development efforts. The great possibilities offered by current technology are pushing users to modify their role as passive consumers, evolving towards being co-producers of the tools they use. An increasing

number of people need to to tailor the software they use to make it better suited to their own needs. Researchers agree that EUD is not a luxury anymore but a necessity, requiring new paradigms and approaches to create systems that comply with the new roles as end users in the software life cycle [58], [3]. Thus, not only researchers, but also industry practitioners are very much interested in the topics addressed in this paper, i.e. in whether and how a meta-design model, supporting the design of systems that enable people to perform EUD, can be reused in different contexts and different application domains.

In literature there are many models and methods to support system design or system evaluation. Unfortunately, some of them are very limited in their application to situations other than those they were developed for, making their transferability very difficult, or even impossible. However, good examples of methods transferred to HCI from other disciplines are also available. Some examples are inspection methods for usability evaluation, which originated from code inspection in software engineering, as well as the formal user testing, called controlled experiments, used in experimental psychology. This paper has described how to transfer a design model to different application domains. The main reasons that make this possible are summarized in the following.

1. The model described in this paper is general enough and, as such, it is valid in several contexts. This is because it has been defined by a consolidated scientific approach: a) starting from experiments; b) abstracting the model; c) validating the model with new experiments and possibly revising the model. The SSW model was presented for the first time in 2002 [6]. Many issues emerged from these experiences, such as: the diversity of users; the importance of designing for universal access and thus of developing software environments and tools customized to various users; the possibility to enable end -users to adapt and refine by themselves the environments and tools they use. These issues have been abstracted in the model, together with other important, common features. Once defined, the model was applied in several other contexts and these new experiments were instrumental for validating the model and for refining it, so that it could be clearer, more general and, thus, more widely applicable. Hence, generality of the model is a basic requirement for transferability.

2. In order to help other people to easily transfer a model to a different context, the model should be well articulated in its main components, in particular, specifying all the key elements on which designers have to focus when transferring the model. This is tacit knowledge for the researchers who developed the model, however it has to be made explicit for others. In previous applications of the SSW, tacit knowledge about the model has been applied. This paper has systematically described the SSW key elements, showing with the reported examples that a clear definition of the key elements is instrumental for facilitating transferability.

3. The model was applied in the design of several interactive systems. Its application was successful because the designers were well aware of the key elements of the model and of other important features that were considered. While reasoning on transferability issues, it was realized that, if one is to support other people in using the SSW model, it is necessary to describe explicitly the transferability process that leads to a successful application of this model, which has been the case in the present paper.

4. The model emphasizes the culture of participation, since it focuses on a meta-design paradigm, in order to enable end users to participate in the design of the tools they use. The culture of participation is gaining momentum in different contexts,

since the technology that makes it possible is available today. Products of the culture of participation are Wikipedia® and, in general, Web 2.0. Some other interesting examples are Learning 2.0, i.e. changing the nature of courses taught in universities by considering teachers and students as co-learners [59], or Electricity 2.0, i.e. the use of Smart Grids that allow consumers to be co-producers of sustainable energy [60]. In all these examples, a meta-design approach, like the one represented by the SSW model, is necessary since people may actively participate in shaping the final products only if they are provided with the basic frameworks to which they may add their own contributions. Thus, the SSW model could inspire meta-design even in contexts not strictly related to software systems.

It is worth noting that while point 4 is specific for the SSW model, the first three points provide useful hints for transferability of any model, emphasizing that the process is facilitated if 1) the model is general, 2) its key elements are well-defined and 3) the transferability process is clearly specified.

It has also been illustrated that, in some cases, transferability might require changes in the model, which lead to its evolution or even to a slightly different formulation. In the authors' opinion, this is very acceptable, especially if the new formulation stresses features that are significant of a large class of problems, so that model generality is still ensured. The SSW model evolved in the HMS model primarily to make more explicit the creative participation of different stakeholders in designing applications focused on social interaction, in which the role of end users is very relevant, and stresses that design and use levels are intertwined rather than distinctive. Working in their own software shaping workshop (SSW), users can switch between different levels and different perspectives. This reinforces the model emphasis on the culture of participation: the user is enabled to modify the system while using it, increasingly blurring the distinction between design time and use time.

ACKNOWLEDGEMENTS

This work was partially supported by the EU through COST Action IC0904 "TwinTide", by the Italian Ministry of University and Research (MIUR) under grant VINCENTE and by Italian Ministry of Economic Development (MISE) under grant LOGIN. The work of Li Zhu was supported by the Initial Training Network "Marie Curie Actions", funded by the FP 7 - People Programme with reference PITN-GA-2008-215446 entitled "DESIRE: Creative Design for Innovation in Science and Technology".

REFERENCES

1. Cockton, G.: Domain Values and Method Transferability: an Initial Framework. 1st European Workshop on HCI Design and Evaluation: The influence of domains, Limassol, Cyprus (2011) 85-90
2. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: Visual Interactive Systems for End-User Development: a Model-based Design Methodology. IEEE T. Syst. Man Cy. A 37(6), 1029-1046 (2007)
3. Ardito, C., Buono, P., Costabile, M.F., Lanzilotti, R., Piccinno, A.: End users as co-designers of their own tools and products. J. Vis. Lang. Comput. 23(2), 78-90 (2012)
4. Fogli, D., Colosio, S., Sacco, M.: Managing accessibility in local e-government websites through end-user development: a case study. Universal Access in the Information Society 9(1), 35-50 (2010)
5. Fogli, D., Parasiliti Provenza, L.: A meta-design approach to the development of e-government services. J. Vis. Lang. Comput. 23(2), 47-62 (2012)
6. Costabile, M.F., Fogli, D., Fresta, G., Mussio, P., Piccinno, A.: Computer Environments for Improving End-User Accessibility. In: Carbonell, N., Stephanidis, C. (eds.) Universal Access Theoretical Perspectives, Practice, and Experience. LNCS, vol. 2615, pp. 129-140. Springer, Berlin / Heidelberg (2003)
7. Costabile, M.F., Fogli, D., Fresta, G., Mussio, P., Piccinno, A.: Building environments for End-User Development and Tailoring. In: IEEE Symposium on Human Centric Computing Languages and Environments, pp. 31-38. IEEE Computer Society (2003)
8. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: End-User Development: the Software Shaping

Workshop Approach. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, vol. 9, pp. 183-205. Springer, Dordrecht, The Netherlands (2006)
9. Costabile, M.F., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: Supporting End Users to Be Co-designers of Their Tools. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) End-User Development. LNCS, vol. 5435, pp. 70-85. Springer, Berlin / Heidelberg, Germany (2009)
10. Sutcliffe, A., Mehandjiev, N.: End-user development (Introduction to Special Issue). Commun. ACM 47(9), 31-32 (2004)
11. Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M.B., Rothermel, G., Shaw, M., Wiedenbeck, S.: The state of the art in end-user software engineering. ACM Comput. Surv. 43(3), 1-44 (2011)
12. Costabile, M.F., Fogli, D., Marcante, A., Piccinno, A.: Supporting Interaction and Co-evolution of Users and Systems. In: International Conference on Advanced Visual Interface, pp. 143-150. ACM Press (2006)
13. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A., Mehandjiev, N.: Meta-design: a manifesto for end-user development. Commun. ACM 47(9), 33-37 (2004)
14. Fischer, G.: End-User Development: From Creating Technologies to Transforming Cultures. In: Dittrich, Y., Burnett, M., Mørch, A., Redmiles, D. (eds.) End-User Development. LNCS, vol. 7897, pp. 217-222. Springer, Berlin Heidelberg (2013)
15. Fischer, G.: Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain-Oriented Design Environments. Automated Software Engineering 5(4), 447-464 (1998)
16. Koehne, B., Redmiles, D., Fischer, G.: Extending the Meta-design Theory: Engaging Participants as Active Contributors in Virtual Worlds. In: Costabile, M.F., Dittrich, Y., Fischer, G., Piccinno, A. (eds.) End-User Development. LNCS, vol. 6654, pp. 264-269. Springer Berlin Heidelberg (2011)
17. Maceli, M., Atwood, M.E.: From Human Crafters to Human Factors to Human Actors and Back Again: Bridging the Design Time – Use Time Divide. In: Costabile, M.F., Dittrich, Y., Fischer, G., Piccinno, A. (eds.) End-User Development. LNCS, vol. 6654, pp. 76-91. Springer Berlin Heidelberg (2011)
18. Costabile, M.F., Fogli, D., Marcante, A., Mussio, P., Parasiliti Provenza, L., Piccinno, A.: Designing Customized and Tailorable Visual Interactive Systems. IJSEKE 18(3), 305-325 (2008)
19. Iverson, K.E.: Notation as a tool of thought. Commun. ACM 23(8), 444-465 (1980)
20. Beyer, H., Holtzblatt, K.: Contextual design: defining customer-centered systems. Morgan Kaufmann, San Francisco (1998)
21. Ardito, C., Barricelli, B.R., Buono, P., Costabile, M.F., Lanzilotti, R., Piccinno, A., Valtolina, S.: An Ontology-Based Approach to Product Customization. In: Costabile, M.F., Dittrich, Y., Fischer, G., Piccinno, A. (eds.) End-User Develpment, vol. 6654, pp. 92-106. Springer, Berlin / Heidelberg (2011)
22. Barricelli, B.R., Mussio, P., Padula, M., Scala, P.L.: TMS for multimodal information processing. Springer Netherlands 54(1), 97-120 (2011)
23. Ardito, C., Buono, P., Costabile, M.F., Lanzilotti, R.: Two different interfaces to visualize patient histories on a PDA. In: 8th conference on Human-computer interaction with mobile devices and services, pp. 37-40. ACM, New York, NY, USA (2006)
24. Costabile, M.F., Fogli, D., Lanzilotti, R., Mussio, P., Piccinno, A.: Supporting Work Practice Through End-User Development Environments. Journal of Organizational and End User Computing 18(4), 43-65 (2006)
25. Ardito, C., Buono, P., Costabile, M.F., Lanzilotti, R., Simeone, A.L.: An Information Visualization Approach to Hospital Shifts Scheduling. In: Jacko, J.A. (ed.) Human-Computer Interaction. Interacting in Various Application Domains. LNCS, vol. 5613, pp. 439-447. Springer, Berlin Heidelberg (2009)
26. Morrison, C., Blackwell, A.: Observing End-User Customisation of Electronic Patient Records. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) End-User Development. LNCS, vol. 5435, pp. 275-284. Springer, Berlin / Heidelberg (2009)
27. Cabitza, F., Simone, C.: WOAD: A Framework to Enable the End-User Development of Coordination-Oriented Functionalities. Journal of Organizational and End User Computing 1-20 (2010)
28. Cabitza, F., Gesso, I., Corna, S.: Tailorable flexibility: Making end-users autonomous in the design of active interfaces. In: IADIS Multi Conference on Computer Science and Information Systems, pp. 53-60. (2011)
29. Ardito, C., Costabile, M.F., De Angeli, A., Lanzilotti, R.: Enriching Archaeological Parks with Contextual Sounds and Mobile Technology. ACM Trans. Comput.-Hum. Interact. 19(4), 1-30 (2012)
30. Ardito, C., Costabile, M.F., Lanzilotti, R.: Gameplay on a multitouch screen to foster learning about historical sites. In: International Conference on Advanced Visual Interfaces (AVI), pp. 75-78. ACM, New York, NY, USA (2010)
31. Ardito, C., Lanzilotti, R., Costabile, M.F., Desolda, G.: Integrating traditional learning and games on large displays: an experimental study. Educational Technology & Society 16(1), 44-56 (2013)
32. Simeone, A.L., Ardito, C.: EUD Software Environments in Cultural Heritage: A Prototype. In: Costabile, M.F., Dittrich, Y., Fischer, G., Piccinno, A. (eds.) End-User Development. LNCS, vol. 6654, pp. 313-318. Springer Berlin / Heidelberg (2011)
33. Edmonds, E., Candy, L., Cox, G., Eisenstein, J., Fischer, G., Hughes, B., Hewett, T.: Individual and/versus social creativity (panel session). 3rd conference on Creativity & Cognition. ACM,

Loughborough, United Kingdom (1999) 36-39
34. Dourish, P.: The Appropriation of Interactive Technologies: Some Lessons from Placeless Documents. Comput. Supported Coop. Work 12(4), 465-490 (2003)
35. Suchman, L.A.: Plans and Situated Actions: The Problem of Human-Machine Communication. Cambridge University Press, New York, NY, USA (1987)
36. Zhu, L., Mussio, P., Barricelli, B.R., Iacob, C.: A habitable space for supporting creative collaboration. Collaborative Technologies and Systems (CTS). IEEE, Chicago (2010) 617-625
37. Zhu, L., Mussio, P., Barricelli, B.R.: Hive-mind space model for creative, collaborative design. In: 1st DESIRE Network Conference on Creativity and Innovation in Design, pp. 121-130. Desire Network (2010)
38. Ardito, C., Barricelli, B.R., Buono, P., Costabile, M.F., Piccinno, A., Valtolina, S., Zhu, L.: Visual Mediation Mechanisms for Collaborative Design and Development. In: Stephanidis, C. (ed.) Universal Access in Human-Computer Interaction. Design for All and eInclusion. LNCS, vol. 6765, pp. 3-11. Springer, Berlin / Heidelberg (2011)
39. Carlile, P.R.: A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development. Organization Science 13(4), 442-455 (2002)
40. Zhu, L.: Cultivating collaborative design: design for evolution. Second Conference on Creativity and Innovation in Design (DESIRE '11). ACM, Eindhoven, Netherlands (2011) 255-266
41. Leuf, B., Cunningham, W.: The Wiki Way: Collaboration and Sharing on the Internet. Addison-Wesley Professional, Boston, MA, USA (2001)
42. Brandt, J., Guo, P.J., Lewenstein, J., Klemmer, S.R.: Opportunistic programming: how rapid ideation and prototyping occur in practice. 4th international workshop on End-user software engineering (EUSE). ACM, Leipzig, Germany (2008) 1-5
43. Schuler, D., Namioka, A.: Participatory Design: Principles and Practices. Lawrence Erlbaum Associates, Inc. (1993)
44. Bødker, S., Grønbæk, K.: Design in action: from prototyping by demonstration to cooperative prototyping. In: Greenbaum, J., Kyng, M. (eds.) Design at work: Cooperative design of computer systems, pp. 197-218. L. Erlbaum Associates Inc., Hillsdale, NJ (1992)
45. Letondal, C., Mackay, W.E.: Participatory programming and the scope of mutual responsibility: balancing scientific, design and software commitment. In: Eighth conference on Participatory design (PDC): Artful integration: interweaving media, materials and practices - Volume 1, pp. 31-41. ACM, New York, NY, USA (2004)
46. Mørch, A.: Three levels of end-user tailoring: customization, integration, and extension. In: Kyng, M., Mathiassen, L. (eds.) Computers and design in context, pp. 51-76. MIT Press (1997)
47. Mackay, W.E.: Triggers and barriers to customizing software. In: SIGCHI conference on Human factors in computing systems: Reaching through technology, pp. 153-160. ACM, New York, NY, USA (1991)
48. Nardi, B.: A Small Matter of Programming: Perspectives on End User Computing. The MIT Press, Cambridge, MA (1993)
49. Mørch, A.I., Stevens, G., Won, M., Klann, M., Dittrich, Y., Wulf, V.: Component-based technologies for end-user development. Commun. ACM 47(9), 59-62 (2004)
50. Burnett, M., Cook, C., Rothermel, G.: End-user software engineering. Commun. ACM 47(9), 53-58 (2004)
51. Brodie, C.B., Hayes, C.C.: DAISY: a decision support design methodology for complex, experience-centered domains. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 32(1), 50-71 (2002)
52. Penner, R.R., Steinmetz, E.S.: Model-based automation of the design of user interfaces to digital control systems. IEEE T. Syst. Man Cy. A 32(1), 41-49 (2002)
53. Fonseca, J.M.C.: W3C Model-Based UI XG Final Report. http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/ (2010)
54. da Silva, P.P.: User Interface Declarative Models and Development Environments: A Survey. In: Palanque, P., Paternò, F. (eds.) Interactive Systems Design, Specification, and Verification. LNCS, vol. 1946, pp. 207-226. Springer, Berlin Heidelberg (2001)
55. Trewin, S., Zimmermann, G., Vanderheiden, G.: Abstract representations as a basis for usable user interfaces. Interacting with Computers 16(3), 477-506 (2004)
56. Kuutti, K.: Activity theory as a potential framework for human-computer interaction research. In: Nardi, B.A. (ed.) Context and Consciousness: Activity Theory and Human Computer Interaction, pp. 17-44. MIT Press, Cambridge, MA, USA (1995)
57. Schön, D.A.: The Reflective Practitioner: How professionals think in action. Basic Books, New York (1983)
58. Fischer, G.: End User Development and Meta-Design: Foundations for Cultures of Participation. Journal of Organizational and End User Computing 22(1), 52-82 (2010)
59. dePaula, R., Fischer, G., Ostwald, J.: Courses as Seeds: Expectations and Realities. In: Dillenbourg, P., Eurelings, A., Hakkarainen, K. (eds.) Proceedings of The European Conference on Computer-Supported Collaborative Learning, pp. 494-501. Maastricht, Netherlands (2001)
60. National Renewable Energy Laboratory's (NREL): Open Energy Information (OpenEI), http://en.openei.org/. Last access on April 2, 2013