# "Each to His Own": Distinguishing Activities, Roles and Artifacts in EUD Practices

**Federico Cabitza, Daniela Fogli and Antonio Piccinno**

**Abstract** End-User Development (EUD) studies how to empower end users (among which, e.g., professionals and organizational workers) to modify, adapt and extend the software systems they daily use, thus coping with the evolving needs of their work organizations and the shop-floor environment. This research area is becoming increasingly important also for the cross fertilization of ideas and approaches that come from the fields of Information Systems and Human-Computer Interaction. However, if one considers the variety of research proposals stemming from this common ground, there is the risk of losing denotational precision of the key terms adopted in the common vocabulary of EUD. To counteract this natural semantic drift, the objective of this paper is to distinguish within three EUD complementary important notions, namely activities, roles, and artifacts, in order to help researchers deepen important phenomena regarding the "meta-design" of systems built to support EUD practices.

**Keywords** End-user development · User task · Meta-design · Intermediary object · Knowledge artifact

F. Cabitza
Università degli Studi di Milano-Bicocca, Milan, Italy
e-mail: cabitza@disco.unimib.it

D. Fogli
Università degli Studi di Brescia, Brescia, Italy
e-mail: fogli@ing.unibs.it

A. Piccinno (✉)
Università degli Studi di Bari, Bari, Italy
e-mail: antonio.piccinno@uniba.it

L. Caporarello et al. (eds.), *Smart Organizations and Smart Artifacts*,
Lecture Notes in Information Systems and Organisation 7,
DOI: 10.1007/978-3-319-07040-7_19,

## 1 Introduction

Humanist studies and disciplines have often provided Information Systems (IS) and Human-Computer Interaction (HCI) researchers with sets of so-called "sensitizing" concepts, that is terms and expressions that "help analysts unpack the social organization of cooperative activities" [1]. However, a natural semantic drift usually occurs when such sensitizing concepts are borrowed by scholars of different disciplines. This has occurred, especially in the HCI field, with regard to the concepts of boundary object [2], community of practice [3], and appropriation [4], as well as to the recent concept of meta-design [5]. In this drift, as also argued in [6], subtle but yet important nuances of the original concepts are either completely lost or get blurred in the description and analysis of social settings; in this way, they fail to inform requirement elicitation and Information Technology (IT) design to their full potential.

The objective of this paper is to shed light on three necessary notions, which could help researchers better analyze subtle but yet important phenomena for a more successful meta-design in End-User Development (EUD) initiatives [7, 8]. Indeed, in 2003, EUD has been defined as "the set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact" [9]. However, if one analyses the variety of proposals in the EUD field, like those included, for example, in the book on EUD [9] or in the proceedings of the so far four editions of the International Symposium on EUD, it is possible to observe that, over the years, such a definition became blurred and too general, due to the possibilities provided by technology today (e.g., the advent of the Web 2.0 and 3.0) that ten years ago people could not anticipate. Indeed, the objective of this paper is to propose a clear distinction in the EUD discourse between complementary articulations around three perspectives: namely, the perspectives of *activities*, *roles* and *artifacts*, according, respectively, to the aim and scope of the EUD practice, to whom is to take advantage of the product of such a practice, and lastly to the role of mediation played by IT artifacts in EUD scenarios. This threefold articulation is based on a series of field studies that the authors have performed in the last years in heterogeneous and unrelated EUD projects, whose main element in common has been the endeavor of adapting the meta-design framework to real communities of practice and practitioners.

The paper is organized as follows. Section 2 presents a classification of EUD activities. In Sect. 3 the roles in EUD practices are discussed, while in Sect. 4 a classification of the artifacts as a result of the different EUD activities is presented. Finally, in Sect. 5 the implications that the three articulations may have on meta-design activities and on the role of meta-designers are discussed.

## 2 Activities

To disentangle EUD articulations, we propose at first to classify EUD into *individual EUD* and *public EUD* (see Fig. 1). Individual EUD encompasses all those activities that lead to the creation, modification or extension of a software artifact for personal use only. Typical examples of individual EUD regard spreadsheet programming [10], where end users create or modify formulas and macros for their own purposes, or scripting environments for statistical computing and graphics, like R and MatLab, by which experts in scientific domains (biology, statistics, geology, etc.) write usually short bunches of software code to analyze and display their data autonomously [11]. Individual EUD is the main research subject of End-User Software Engineering (EUSE) [12], which proposes a variety of methods for requirement analysis and specification, system design and reuse, verification and testing, code debugging, specifically devoted to non-professional software developers.

However, in many situations, single end users either program or configure software artifacts that are used by (or also by) *other* people, as in the case of multi-tiered proxy design problems [13, 14]. Usually these people are colleagues and co-workers (as in the case of the Electronic Patient Record in [15]), but also people working in other departments, contexts, and communities (as in the case of e-government in [16]): In this case, we speak of public EUD, since the outcome of the EUD activity is aimed at being shared and publicly available to others than the end user involved in the programming activity. The main difference between public and individual EUD is then the explicit intention behind the programming effort: either making something intended to be shared or not, respectively.

Public EUD can be further specialized into *inward EUD* and *outward EUD*. In the former case, the people carrying out any EUD activity work for a community they also belong to, with or without the intention to build an artifact that could be adopted also in other, possibly different, settings. Although this cannot be a priori excluded, EUD activities are intended to support members of small teams and groups of people sharing sets of conventions, assumptions and practices, i.e., in many cases what are called "communities of practice" [17]. In these settings many things can be given for granted and computational support is intentionally adjusted in a "quick and dirty" fashion to achieve effectiveness and flexibility, rather than maintainability and transferability. An example is a "pipe" in Yahoo!Pipes: this is usually developed for personal use, but it can also (either intentionally or unintentionally) be shared among the Yahoo!Pipes community. In the outward EUD case, conversely, the quality of the software artifacts is more likely to be purposely pursued, as the EUD activity is intentionally aimed at building and improving tools that are to be used across different communities or, even, in other communities.

In the case of inward EUD, who undertakes EUD tasks is usually denoted with a number of different names, like "power user", "gardener" or "local developer" [18]; these terms are usually used to indicate someone who, belonging to a given community, works for the proficiency of the community itself, in virtue of a deep

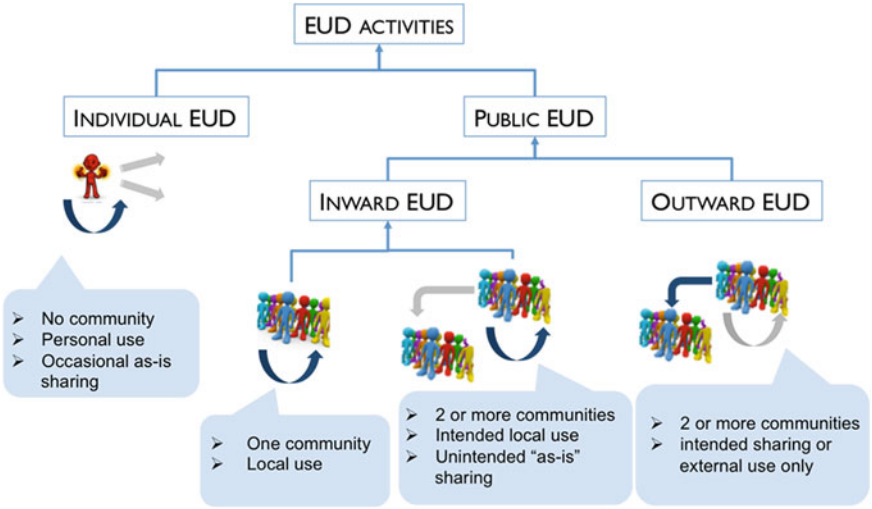


**Fig. 1** EUD activities articulation

and often tacit knowledge of the characteristics and skills of its members. For example, in the case of the Electronic Patient Record discussed in [15], the head physician is called to visually compose the software environment, i.e., the electronic patient record that will be used in her/his ward by her/his colleagues and co-workers. In [19], the authors report of a system in which doctors could create simple rules so that relevant information displayed in their medical records could be highlighted according to the context. Another example is in the archaeological context: in [20], it is described how professional guides use different software solutions that allow them to create personal information spaces, which could be shared with other colleagues through annotation mechanisms.

Conversely, in outward EUD, at least two communities are involved and there is no guarantee that those who carry out EUD activities will also take advantage of the product of these activities; this case has been investigated to a lesser extent by the EUD community, but, nevertheless, can occur in complex and important domains. For instance, in [21] the authors report how civil servants, acting as non-professional software developers, are called to create e-government services for citizens; in [13] it is described how, in a similar way, mobile user interfaces for disabled patients can be easily developed by caregivers (parents or assistants) by means of a script-based system; in [22], it is reported and discussed how editorial staff members of a Web shop portal may be asked to personalize systems for shop owners, and, finally, in the archaeological context mentioned above [20, 23], each guide can retrieve and compose, via a desktop application, the material for his/her personal information space to be shown to a group of visitors during a visit of an archaeological park.

This classification, far from being a rigid taxonomy of EUD practices, hints rather at a spectrum of possible uses (see Fig. 1) and is provided for its role in pointing to specific design implications that will be discussed in Sect. 5.

## 3 Roles

Distinguishing between kinds of activities allows also to consider more precisely scopes and roles involved in those activities. In particular, in addition to the oft-cited roles of the *end user* and *meta-designer,* we propose to consider also the roles of *domain developer* and *maieuta-designer*, according to the task these people are supposed to undertake in an EUD process of IT artifact construction.

As widely known, the end user is a passive user of the IT artifact and consumer of its products and services.

We propose the term "domain developer" to subsume all those roles that are in charge of carrying out EUD activities, like the above mentioned "power user", "local developer", "gardener", "end-user developer" [14], "bricolant bricoleur" [24]. Therefore, a domain developer is a domain expert actively involved in the so-called *meta-task* of improving the system used in the domain-specific task: such a task is "meta" in that it is aimed at creating better artifacts for the main tasks in the work domain at hand.

Likewise, the meta-designer is someone involved in the design of the EUD environment and tools by which domain developers can build their own artifacts. Therefore, s/he is usually a professional software developer in charge of creating the *technical* conditions for EUD practice, according to a vision of meta-design as a two-phase process [7, 25]: The former consisting of designing the design environment, the latter consisting of designing the artifacts for end users using the design environment.

On the other hand, the maieuta-designer specializes some of the activities that previous literature contributions assigned to the meta-designer and which are more concerned with the establishment and conservation of the most favorable social conditions for empowering and motivating users in shaping their tools at use time [5, 8]. This role actually encompasses and subsumes those that are involved in the task of supporting the meta-task of the domain developers, that is of having the domain experts (playing the role of domain developers) internalize the design culture and the technical notions necessary for the meta-task of artifact development. The maieuta-designer is therefore supposed to facilitate the evolution of single users from being passive end users of their tools to become domain developers, that is domain experts capable to develop their own tools and make them more fit to their settings, or at least to empower and help end users appropriate their IT artifacts more actively and consciously, so that they can commit themselves in improving the artifact e.g., by simply reporting shortcomings and system faults, and expressing due modifications and appreciated improvements to whom it concerns or is able to intervene (i.e., the domain developers or the IT professionals, if

available). For this reason we call such a designer a maieuta-designer, partly in analogy with the Socratic method of getting people acquire notions, motivations and self-confidence to undertake challenging tasks by themselves, and partly in clear assonance with the term meta-designer, of which it is a specialization more oriented to work practice and EUD activities than to IT design and development [24]. The maieuta-designer role is also in line with a new and more recent vision of EUD that considers it (and meta-design) as one of the foundations of the cultures of participation rather than a mere technical instrument [8].

Domain developer and maieuta-designer are just new roles that we have introduced to specify the activities of the end user and meta-designer in EUD practice (see Fig. 2). However, nothing prevents that the same person plays different roles at different degrees: for example, especially in individual EUD, the end user actually becomes, at some time, a domain developer. Similarly, the roles of meta-designer and maieuta-designer could be played by a software engineer and by a HCI expert respectively, as in the case described in [26], or both roles may be played by a professional software developer as in [13]. In the same vein, the maieuta-designer could be occasionally just the most passionate one of the practitioners involved in an EUD initiative, who tries to convince his/her colleagues to join the initiative as well and have an active stake in the continuous improvement of the IT artifacts that are in their partial or total control.

Indeed, as Fig. 2 shows, these four roles are also aimed at representing a sort of continuum in the attitude towards the IT artifact, from the less active one, i.e., the end user that just uses the IT artifact and occasionally gives feedback to the person who is officially accountable for its quality, often on a professional basis (i.e., the meta-designer). Compliant with the EUD tenets, our categorization does not force domain analysis to fit actors into narrow boxes that do not reflect the complexity of real work settings; quite the opposite we recognize such a complexity considering that no wall should be established between roles, and that responsibilities, although clear at any given time, can change as the project unfolds over time and contributions are given on a voluntary basis irrespective of the intended planning. The unpredictability and necessary openness of EUD projects is what makes them substantially different from traditional software engineering projects and urges for role and activity models that could cope with situated processes of IT appropriation and collaborative development of the technology in a community context.

## 4 Artifacts

The classification proposed above, which distinguishes between individual EUD and public EUD, as well as between inward and outward EUD, allows also to distinguish in a finer-grained manner the IT artifacts used by the roles involved in those activities.

In particular, it is useful to distinguish between: *personal artifacts*, which are used by single users that employ the EUD environment for their own purposes, as

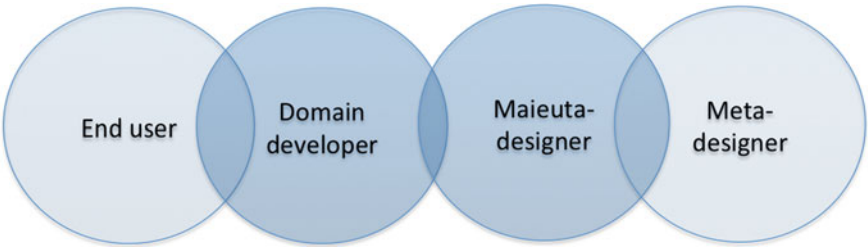


**Fig. 2** Roles in EUD practices

in the case of spreadsheet programming; and *intermediary objects*. These are objects that are shared, exchanged and circulated among members of networks and communities to mediate their interactions [27]; in so doing, they represent the intermediate steps of any design or meta-design task [28], i.e., "the traces as well as the outputs of a collaborative transformational process" [29] and thus support the continuous process by which they and other objects evolve over time. Personal artifacts and intermediary objects mirror the EUD activities that they support: individual and public EUD, respectively. Intermediary objects can be further distinguished in *knowledge artifacts* and *boundary objects*, as extremes of a continuous range that do not only cross boundaries but also contribute in shaping them [29, 30]. Boundary objects is the notion introduced by Bowker and Star in [31] to account for those artifacts that enable a sort of standardized and effectively simplified communication and coordination between members of different communities of practice; knowledge artifacts, on the other hand, are artifacts that enable and support learning and innovation within a specific community of practice, that is processes of knowledge acquisition, circulation and creation among its members [32]. This spectrum of EUD artifacts is illustrated in Fig. 3.

In the EUD literature, boundary objects are often recognized between the community of end users and the community of the IT professionals (i.e., meta- and maieuta-designers) involved in the digitization process (e.g. [33]). Although this can be perfectly the case, we should always remember that, in a real EUD scenario, IT professionals should be present only at the inception of a digitization project, as EUD is ultimately aimed at making end users autonomous in the long run. For this reason, in standard use, speaking of EUD software artifacts as intermediary objects seems more appropriate. That notwithstanding, either boundary objects or knowledge artifacts, as particular instances of intermediary objects, cannot be rigidly associated with either Inward or Outward EUD activities: it is a matter of analysis to understand what roles an IT artifact is playing in an organizational domain and understand how to support it computationally. Indeed, on one hand, inward EUD addresses IT artifacts that often play the role of knowledge artifact; moreover, hosting or being the objective of inward EUD activities make those IT artifacts become also knowledge artifacts in the process itself of development, as these latter end up by "mirroring and reflecting" how the community (or its

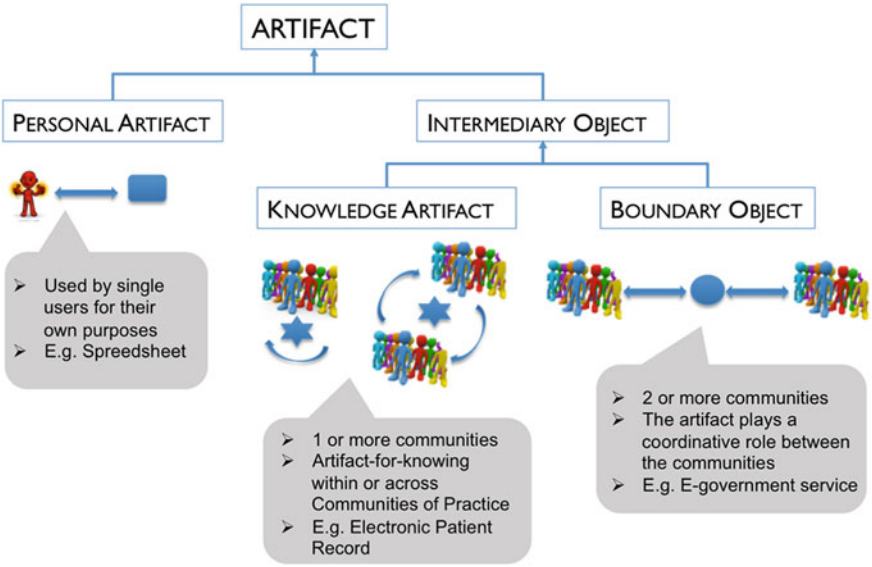


**Fig. 3** EUD artifacts articulation

domain developers) has improved both the work tasks and the meta-tasks that characterize the community itself. This is for example the case of the Electronic Patient Record in the medical domain [15, 19, 23], which represents a shared artifact within a hospital ward and among different wards, useful for accumulating and sharing knowledge about each patient. In outward activities IT artifacts can "fertilize" different communities, by enabling activities that require some learning by the members of the "receiving" community: therefore also in this case it is possible to speak of knowledge artifacts "across communities of practice" [32]. On the other hand, outward EUD always encompasses the transfer of more or less full-fledged IT artifacts from one community to another, but this does not necessarily imply that these latter will play any coordinative role between these two communities, thus being boundary objects as Bowker and Star defined them originally [31]. However, one of the main reasons why a community could want to develop tools to be given to other communities is to achieve a better communication and alignment of meanings, purposes and activities, so as to make those tools effective boundary objects: for instance, an administrative office could provide a commercial office with a partially precompiled spreadsheet to have commercial agents fill in expense accounts more accurately and completely, as reported in [34]. Another example is in the e-government domain, where a civil servant may create the description of an e-government service, which gives rise to the automatic generation of both the web pages to be used by citizens to apply for the service, and of the web pages to be used by administrative employees to manage citizens' requests [16].

Once again, our point is that distinguishing between different activities, roles and artifacts is not a nominalist effort, but rather an analytic stance that allows for detecting nuances that could call for different design approaches, quality requirements and EUD solutions.

## 5 Discussion and Conclusion

In this paper we proposed a threefold "activity, role and artifact" perspective in EUD, in order to address complementary and mutually affecting components of an EUD project. As said above, our effort is not merely taxonomic, but rather oriented to the situated practices of requirement analysis and design. Indeed, a fine-grained articulation of roles can help in understanding how to deploy efficient training programs and apply effective rewarding mechanisms; for instance, distinguishing between end-user, domain developer, and meta- and maieuta-designer, according to the level of involvement in the process of artifact production, can help in detecting different tasks to be supported in different ways and hence in shedding light on specific meta-design principles. On the other hand, distinguishing what kind of preexisting artifact has to be digitized in an organizational domain, that is focusing on what main role a traditional artifact is playing, which has to be substituted by a software application or IT artifact, would help designers invest on more critical functionalities that are typically necessary in one case, but redundant if not detrimental in the other one, or in detecting a palette of reusable off-the-shelf EUD components to offer to domain developers. In the same light, the analysis we outlined above on the distinction between individual and public EUD suggests to reconsider meta-design priorities and the meta-designer's role.

To this aim, we find it interesting to recall the seminal work of Grudin [35], who discusses the different emphasis put on utility and usability of software systems in different development contexts: whilst in in-house and internal system development emphasis is rightly put more on utility since IT artifacts are built around their intended functionalities, in commercial projects it is conversely usability the most important characteristic, as one of the priorities is to facilitate system acceptance by users and therefore increase the likelihood of success of the digitization process. This tension between utility and usability has also influenced different approaches to IT artifact development that have been pursued in the IS and HCI communities over the years respectively. Nowadays, the cross-fertilization of these two research fields can provide insights for dealing with this controversial relation between usability and utility that in the EUD discourse revives as a primary concern and can find, hopefully, an effective solution.

On the one hand, if one considers individual EUD and public/inward EUD, emphasis should be mainly put on utility: indeed, EUD activities encompass system adaptation and extension to increase effectiveness of the individual user and/or of the whole community. Usually this is achieved through development techniques that are very close to traditional programming languages, such as

macro or script development, component-based development and programming by examples. This requires that domain experts are trained (or even self-trained) in programming methods and languages, so as to become what we have therefore called "domain developers", that is domain experts that develop IT artifacts for their own domain or setting. More specifically, in individual EUD, end users and domain developers coincide as people who are in control of modifying the IT artifact for their own purpose.

Conversely, in public/outward EUD, domain developers modify the IT artifact by constantly taking into account the requests of other end users, for the sake of the whole community's advantage, a community they often do not belong to.

As a consequence, to support the meta-task of domain developers in individual EUD and public/inward EUD, meta-designers must focus on the design of EUD tools and infrastructures for communication within the community, whilst the maieuta-designer must focus on the proper training of the domain developers, and on managing the risk and the impact that the system under evolution may have on the organization and its work practices.

On the other hand, in public/outward EUD, artifacts must be designed more carefully for at least two reasons: first, because these artifacts will be used by people that did not participate in the development process and could find contacting the developers very hard, if not impossible at all. In fact, an EUD artifact is not guaranteed by any commercial company, nor a help desk exists to troubleshoot its problems or provide guidance about its proper use. Secondly, it is possible that other non-professional software developers will have to adjust the artifacts to make them more suitable and fit to the community of end users where these are eventually adopted. This means that public/outward EUD requires a greater emphasis on usability: not only tools supporting domain developers must fit their characteristics, skills and background, but also the artifacts created for end users by the domain developers must be usable as well. Thus, in this case, EUD techniques must be both informed by domain-specific concepts and oriented toward the support of daily work practices: for instance, EUD environments in e-government [21] and medical projects [36] adopt the metaphors of the form-based interaction and of the active document because these recalls the usual ways work is accomplished in those domains. Furthermore, the activity of domain developers consists of creating software artifacts for people that belong to a different community; thus, proper mechanisms for making artifact creation easier and code generation transparent, that is free of unnecessary implementation details or language-specific technicalities, must be defined, along with procedures that guarantee the creation of usable artifacts [37]. Both aspects still regard the meta-task of domain developers, which should be supported no more with training in programming, but rather with user-friendly and visually engaging EUD systems [38], along with proper incentive and rewarding mechanisms. Therefore, in public/outward EUD, the meta-designer must pay more attention on easy-to-use EUD techniques and automatic code generation mechanisms (for example through meta-modeling [39]), whilst the maieuta-designer must focus on motivation strategies.

In light of the considerations mentioned above, novel meta-design guidelines emerge and should be refined and put to the "test of life". Therefore, our future work will be aimed at extending the current proposals on meta-design and making them more concretely applicable in the challenging context of the communities of practice for the creation of supportive EUD tools and infrastructures that can evolve more easily along with the needs and objectives of the members of those communities.

## References


1. Hughes, J.A., Randall, D., Shapiro, D.: Faltering from ethnography to design. In: ACM Conference on Computer-Supported Cooperative Work, pp. 115–122. ACM Press, New York (1992)
2. Star, S.L.: This is not a boundary object: reflections on the origin of a concept. Sci. Technol. Human Values **35**(5), 601–617 (2010)
3. Duguid, P.: Prologue: community of practice then and now. In: Amin, A., Roberts, J. (eds.) Community, Economic Creativity, and Organization, pp. 1–10. Oxford University Press, Oxford (2008)
4. Dix, A.: Designing for appropriation. In: 21st British HCI Group Annual Conference on People and Computers: HCI…but not as we know it, vol. 2, pp. 27–30. British Computer Society, Swinton (2007)
5. Fischer, G., Giaccardi, E.: Meta-design: a framework for the future of end user development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, vol. 9, pp. 427–457. Springer, Dordrecht (2006)
6. Alter, S.: Work systems and IT artifacts—does the definition matter? Commun. Assoc. Inf. Syst. **17**, 299–313 (2006)
7. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: A meta-design approach to End-User Development. In: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 308–310. IEEE Computer Society (2005)
8. Fischer, G.: End user development and meta-design: foundations for cultures of participation. J Organ. End User Comput. **22**(1), 52–82 (2010)
9. Lieberman, H., Paternò, F., Wulf, V. (eds.): End User Development. Springer, Dordrecht (2006)
10. Burnett, M., Rothermel, G., Cook, C.: An integrated software engineering approach for end-user programmers. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, vol. 9, pp. 87–113. Springer, Netherlands (2006)
11. Letondal, C.: Participatory programming: developing programmable bioinformatics tools for end-users. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 207–242. Springer, Dordrecht (2006)
12. 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)
13. Carmien, S., Dawe, M., Fischer, G., Gorman, A., Kintsch, A., Sullivan Jr, J.F.: Socio-technical environments supporting people with cognitive disabilities using public transportation. ACM Trans. Comput. Hum. Inter. **12**(2), 233–262 (2005)
14. Fogli, D., Piccinno, A.: Co-evolution of end-user developers and systems in multi-tiered proxy design problems. In: Dittrich, Y., Burnett, M., Mørch, A., Redmiles, D. (eds.) End-User Development. LNCS, vol. 7897, pp. 153–168. Springer, Berlin (2013)

15. 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)
16. Fogli, D.: Towards a new work practice in the development of e–government applications. Electron. Gov. Inter. J. **10**(3), 238–258 (2013)
17. Wenger, E., McDermott, R.A., Snyder, W.: Cultivating Communities of Practice: A Guide to Managing Knowledge. Harvard Business Press, Boston (2002)
18. Gantt, M., Nardi, B.A.: Gardeners and gurus: patterns of cooperation among CAD users. In: ACM Conference on Human Factors in Computing Systems (CHI), pp. 107–117. ACM, New York, NY, USA (1992)
19. Cabitza, F., Simone, C.: Affording mechanisms: an integrated view of coordination and knowledge management. Comput. Support. Coop. Work (CSCW) **21**(2–3), 227–260 (2012)
20. Ardito, C., Bottoni, P., Costabile, M.F., Desolda, G., Matera, M., Piccinno, A., Picozzi, M.: Enabling end users to create, annotate and share personal information spaces. In: Dittrich, Y., Burnett, M., Mørch, A., Redmiles, D. (eds.) End-User Development. LNCS, vol. 7897, pp. 40–55. Springer, Berlin (2013)
21. Fogli, D., Provenza, L.P.: A meta-design approach to the development of e-government services. J. Vis. Lang. Comput. **23**(2), 47–62 (2012)
22. 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 (2011)
23. Ardito, C., Buono, P., Costabile, M.F., Lanzilotti, R., Piccinno, A., Zhu, L.: On the transferability of a meta-design model supporting End-User Development. Univ. Access Inf. Soc. J. (UAIS) (in print)
24. Cabitza, F., Simone, C.: Building socially embedded technologies: implications on design. In: Randall, D., Schmidt, K., Wulf, V. (eds.) Designing Socially Embedded Technologies: A European Challenge. Springer, Berlin (in print)
25. Sutcliffe, A., Mehandjiev, N.: End-user development (Introduction to Special Issue). Commun. ACM **47**(9), 31–32 (2004)
26. Costabile, M.F., Fogli, D., Marcante, A., Mussio, P., Provenza, L.P., Piccinno, A.: Designing customized and tailorable visual interactive systems. Inter. J. Softw. Eng. Knowl. Eng. **18**(3), 305–325 (2008)
27. Vinck, D., Blanco, E.: Everyday Engineering: an Ethnography of Design And Innovation. MIT Press, Cambridge (2003)
28. Boujut, J.-F., Blanco, E.: Intermediary objects as a means to foster co-operation in engineering design. Comput. Support. Coop. Work (CSCW) **12**(2), 205–219 (2003)
29. Lee, C.P.: Boundary negotiating artifacts: unbinding the routine of boundary objects and embracing chaos in collaborative work. Comput. Support. Coop. Work (CSCW) **16**(3), 307–339 (2007)
30. Cabitza, F.: At the boundary of communities and roles: boundary objects and knowledge artifacts as complementary resources for the design of information systems. In: Mola, L., Pennarola, F., Za, S. (eds.) From Information to Smart Society: Environment, Politics and Economics. LNISO. Springer, Berlin (in print)
31. Bowker, G.C., Star, S.L.: Sorting Things Out: Classification and Its Consequences. MIT Press, London (1999)
32. Cabitza, F., Colombo, G., Simone, C.: Leveraging underspecification in knowledge artifacts to foster collaborative activities in professional communities. Int. J. Hum. Comput. Stud. **71**(1), 24–45 (2013)
33. Hess, J., Reuter, C., Pipek, V., Wulf, V.: Supporting end-user articulations in evolving business processes: a case study to explore intuitive notations and interaction designs. Inter. J. Coop. Inf. Syst. **21**(4), 263–296 (2012)
34. Batini, C., Barone, D., Cabitza, F., Grega, S.: A data quality methodology for heterogeneous data. Inter. J. Database Manag. Syst. (IJDMS) **3**(1), 60–79 (2011)

35. Grudin, J.: Utility and usability: research issues and development contexts. Interact. Comput. **4**(2), 209–217 (1992)
36. Cabitza, F., Simone, C.: WOAD: a framework to enable the end-user development of coordination-oriented functionalities. J. Organ. End User Comput. **22**(2), 1–20 (2010)
37. Fogli, D., Piccinno, A.: Enabling domain experts to develop usable software artifacts. In: Spagnoletti, P. (ed.) Organizational Change and Information Systems. LNISO, vol. 2, pp. 419–428. Springer, Berlin (2013)
38. Cabitza, F., Gesso, I., Simone, C.: Providing end-users with a visual editor to make their electronic documents active. In: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 171–174. IEEE Computer Society (2012)
39. Fogli, D., Provenza, L.P.: End-user development of e-government services through meta-modeling. In: Costabile, M.F., Dittrich, Y., Fischer, G., Piccinno, A. (eds.) End-User Development. LNCS, vol. 6654, pp. 107–122. Springer, Berlin, (2011)