Scenario

Innovation Happens Elsewhere, but Where Does Design Happen?

Considerations on Design and Participatory Processes in Emerging Information Technologies

Giacomo Poderi

Abstract By taking as departing point the convergence of emerging technologies and their related production practices, this work reflects on design and participatory processes. This contribution tries to highlight how these processes changed from the traditional Information and Software Systems (ISS) context to the emerging one, where technologies are mainly characterized by decentralized and open-ended processes. The paper presents the traditional conception of design as a linear process and problem-solving endeavour and the role that users' participation has within this frame. It then moves to focus on how design for emergent technologies extends beyond the 'production' phase and on how the distribution of (users-)participants to these technologies intertwine with the endeavour of designing them.

Keywords designing; information and software system; users participation; distribution; continuity.

Introduction

In 2005, two software engineers working at Sun Microsystems wrote the influential book *Innovation happens elsewhere: open source as business strategy* (Goldman and Gabriel 2005). The book primarily targeted practitioners and reflected on the Free and Open Source Software (FOSS) paradigm to argue that a new way of innovating emerged in contemporary society. Innovation processes can happen, and indeed do so in FOSS, without the need of keeping secret the organization's knowledge related to innovative products and confining it within the boundaries of internal Research and Development (R&D) departments. According to the authors, in the case of the production of FOSS programmes 'innovation happens elsewhere' and, more importantly, this new way of innovating can be found in other areas of technology production. Thus, according to them, a

TECNOSCIENZA
Italian Journal of Science & Technology Studies
Volume 3(1) pp. 63-72- ISSN 2038-3460
http://www.tecnoscienza.net

T/S TECNOSCIENZA Italian Journal of Science & Technology Studies

phenomenon worth being observed. Translating by analogy the key argument by Goldman and Gabriel into the focus of this contribution, I suggest that something similar is happening for the concept of design in the domain of Information and Software Systems (ISS), and yet again in connection with aspects that characterise the FOSS paradigm and similar 'participatory technologies'.

Starting from the studies of the electrification of London (Bijker and Law 1994) and the development of modern bicycles, bakelites and light bulbs (Bijker 1995), to the ones of domestic technologies (MacKenzie and Wajcman 1985) and users' led innovations (Oudshoorn and Pinch 2003), Science and Technology Studies (STS) have a sound tradition in opening the 'black-box' of technological production and in studying their 'becoming', broadly understood. However, the emergence of new, participatory technologies of knowledge exchange, information access, and content production which are primarily associated with Internet and new media blurred the traditional relationship between production and consumption, producers and users, design and use. Social networking platforms (e.g. Facebook, LinkedIn), media sharing sites (e.g. Flickr, YouTube), blogging software (e.g. Wordpress), crowdsourcing platforms (e.g. Wikipedia) and the whole FOSS paradigm all bring about new forms of design, development, appropriation and use that crucially differentiate[1] their come into being from the one that traditionally interested the STS community.

This contribution provides a look at emerging practices in the area of ISS and specifically connect emergent technologies paradigms to the processes of design, in general, and of participation, in particular. This work outlines the main differences characterising these concepts in their traditional understanding and in their emergent ones.

1. Solving the requirement problem in traditional Information and Software System

> The hardest single part of building a software system is deciding what to build […] No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later. (Brooks 1987, 12)

In ISS, the process of design is one that has often escaped formal, clear and shared definitions, mainly for two reasons: (i) formal definitions, typical of rationalist approaches, are rarely satisfactory when the actual design, the design in practice, is carried out; (ii) the application domain of this human activity has rapidly and constantly changed over the past few decades (Greenbaum and Kyng 1991). However, as generally understood, the design of information or software

[1] For an overview of these differences see the works of Napoli (2010) and Bruns (2008).

systems implies the endeavour of deciding, at a relatively detailed level, 'what to produce'. In other words, design can be seen as an interaction between understanding what is needed in the context of future technology deployment and creating an artefact which satisfies those needs (Winograd and Flores 1986, 4). It is an activity usually performed by professional figures, namely the software (or system) designers, who operate within the phases of formal development projects. Here, design is just one phase which has the goal of 'solving the requirements problem'. That is to: (i) identify as objectively as possible the features that the final artefact will have, (ii) improve upon the departing status quo, (iii) take into consideration the needs of all the actors involved, and (iv) defining a suitable mediation amongst these needs (Sommerville 1995).

Depending on the specific development methodology used for the system production, the design phase has its specific position and role. For instance, the Waterfall Model is a linear and sequential design process for system development firstly formalised in Royce (1970) and, until recently, widely adopted in one of its many versions. The typical incarnation of the model has seven phases: requirements specification, design, implementation, integration, testing and debugging, installation, maintenance. During the first phase, designers evaluate whether or not the system is feasible, investigate the work-site and the activities taking place there to understand what 'is needed' and they formally define the features of the new system through the requirements specification. In the proper design phase, the requirements are modelled into a detailed blueprint that can be developed into an artefact during the implementation phase.

This model has the merit of clearly identifying the logical elements of the design process. However, although it enjoyed wide adoption since the 70s, the severity of the consecutive phases and the reliance on the requirements that are specified during the early stage, pose limitations to it and make it difficult to adopt it for contemporary software development. Indeed, it heavily relies on the assumptions that requirements are easily identifiable, remain constant throughout the whole development cycle and that they are decomposable in problems and solutions (Avison and Fitzgerald 2003).

Several development methodologies exist which attempt to mitigate the limits of the Waterfall Model: agile development (Larman and Basili 2003), rapid application development (Beynon-Davies et al. 1999), extreme programming (Beck, 2000), spiral model (Sommerville 1995). However, from a critical standpoint, these methods do nothing more than recursively repeating, or combining in different ways, the steps of the Waterfall Model in the attempt of introducing some mechanisms that are able to fine tuning the initial design and the process of inscribing it into the final artefact. As rationalist approaches, they all share the same fundamental bias: the idea that problems are identifiable, definable and solvable through analytical steps and engineered procedures.

Starting from the late 60s, the idea that the involvement of the (future) users of a new technological artefact would be beneficial both for the technology and for the users, increasingly gained acceptance also in ISS development, becaming a fundamental prerequisite for any development effort. In IS, the term "user par-

ticipation" refers to the involvement of the future end-users into the design and implementation processes of the system they will use. Commonly recognised benefits of users participation are: (i) an increased adoption and diffusion of the system; (ii) an higher system quality in terms of a more accurate understanding of user requirements; (iii) a better and more efficient use of the system deriving from the possibility for participants to understand it 'from within'. The key process at the basis of these benefits it is the fostering of mutual learning between the people who will use the system and the ones who are producing it (Greenbaum 1993; Kensing and Blomberg 1998). Often, designers and developers are highly trained and skilled in creating technically valid artefacts but they lack proper understanding of the working domain the artefact will be used for. Vice-versa users well understand what kind of working practices the new system should adapt to and have sound knowledge of the working domain, however they cannot grasp the potential and limits of system development, therefore they are not always clear about what they need and what they can expect (Bødker et al. 2004).

Mutual learning is supposed to mitigate the distance between these two groups and, thus, to help solving the fundamental design problem that any development effort faces: deciding what to build. Several different techniques such as future workshops, organizational games, contextual inquiries and ethnographic approaches, emerged over the years in the attempt of involving users in the design process and helping them to articulate their needs and ideas in a way that designers could understand and act upon (Schuler and Namioka 1993).

2. Emerging practices for emerging technologies: distributed participation and continuously designed projects

> Design as a continuing process that goes on after the formal end of the software development project is, of course, 'old news'. [...] The 'new news' is, that this is where much of the action is today, and it is a much more complex and diverse scene than it was ten years ago. (Dittrich 2002, 225)

The previous outline of designing poorly fits the 'come into being' of the participatory technologies mentioned in the introduction. They differ from traditional technology production at least for two related aspects: (i) they blur the boundaries between production and use of the artefact, and (ii) they imply the distribution at the spatial, organizational and temporal level of the socio-technical assemblages[2] that are associated with them. These assemblages portray decentralized and open-ended organization of work together with bottom-up and unpredictable innovation processes. Traditional boundaries that were clearly identifiable amongst the parties and processes of producing, adopting and using

[2] "Assemblage" is used to indicate "associations of humans and non-humans" as proposed by Latour (1987).

a technological artefact are difficult to distinguish. This is particularly true for the relationships production/use and producer/consumer, indeed "the production value chain is transformed to the point of being entirely unrecognisable - in the absence of producers, distributors, or consumers, and the presence of a seemingly endless string of users acting incrementally as content producers by gradually extending and improving the information present in the information common" (Bruns 2008, 21). It is difficult to decide whether writing an encyclopedic article through a crowdsourcing platform, such as Wikipedia, should qualify as using it or contributing to its production. Similarly, it is difficult to decide whether to establish connections with other users on a social networking site can be considered as using the technology or contributing to the creation of that 'social network' which the technology was meant to be.

In these participatory technologies, designing processes take place in a context where mediation with actual, rather than projected, use is unavoidable: the 'solution to the requirements problem' is only sketched (or attempted) before actual use starts. The largest part of the 'problem' – e.g. understanding what should be built and deciding how to build it – is tackled during the actual use of the technological artefact. The idea of completing the development of the artefact, before it is officially deployed for wide use, is abandoned and substituted by the acknowledgement that the artefact will undergo improvements, changes and further development for as long as there is enough interest around it. Approaches such as continuing design-in-use (Henderson and Kyng 1991), continuous design and redesign (Jones 1983), unfinished design (Tonkinwise 2003) try to tackle this challenge in different ways while sharing similar roots and goals: they all acknowledge the impossibility of satisfactorily anticipating future users' practices or to provide a durable vision to inscribe in the artefacts, and they strive to provide highly flexible development processes for these artefacts. As such, all these approaches came to recognize the relevance that the project's infrastructure acquires in the logic of continuing design in use.

According to Star (1999, 380) it is possible to think of an infrastructure as a system of substrates that is, by definition, invisible and part of the background for other kind of works[3]. However, as Bowker and Star (2000) highlighted, the fact that this infrastructure operates in the background, it does not imply neutrality in respect to the activities performed with it. For instance, classification systems embed important decisions relating to what attributes of an object are relevant, thus worth being included in the classification, and what not. The classified attributes can be remembered and acted upon, the non classified ones are lost and forgotten. Even in the case of design in use, an infrastructure cannot be considered neutral in relation to the activity of continuing design in use. Therefore, deciding how to build it becomes of pertinence of a designing interest. It should be 'designed to allow (re-)design after the initial design took place', to paraphrase what Ehn (2008) refers to in terms of meta-design.

[3] E.g. the set of tools, rules, norms that allow the fulfillment of other activities.

The infrastructure is also the 'locus' where distributed participation happens and manifests: it makes possible for people to participate from different locations, to engage in heterogeneous working areas while supporting heterogeneity of skills, tasks, roles and activities (Gumm, 2006). In Suchman's words (2002, 96), it promotes "a dense and differentiated layering of people, activities and things, each operating within a limited sphere of knowing and acting". Moreover, it allows them to collaborate without the need to share the same-time interaction. As developers, designers and users now share the same infrastructure, the resulting general distribution of actors affects both the ones who actively engage into the development of the artefact (Farshchian and Divitini 1999) and the ones who constitute the contextual environment in relation to which design decisions are taken (Martin et al. 2007; Iivari 2009).

FOSS development provides a paradigmatic case both for the continuity of design and distributedness of participation. On the one hand, the key tenet "release early, release often" (Raymond 1999) that characterises its development and release cycles, implies that from the very inception of the software project, as soon as the artefact reaches a minimal yet usable status, this is released for public use and testing. From then on, development and use of the software can proceed together for the whole life span of the project thanks to a complex system of parallel development branches, feedback practices and 'release management' (Michlmayr et al. 2007). Therefore, all the logical elements of the designing process, as outlined in the previous section, are no longer sequentially aligned and iterated, but they overlap each other and are continuously enacted: FOSS assemblages never cease to generate bug reports and fixes, to receive and evaluate features requests, to extend old functionalities and add new ones, in other words, to design and re-design the software, while keeping it usable and used by its users (Gasser et al. 2003). On the other hand, participants collaborate through a system of heterogeneous tools and communication channels, where each tool is associated with a specific activity and each channel is used for specific kind of discussions. For instance, while system evaluation happens through bug reporting on the bug-tracker, the implementation of new features is done on the Version Control System. Similarly, while issues that are traditionally open to wide debate are discussed on dedicated mailing lists or on Internet forums, other matters that require quicker and more direct interactions are discussed on media such as Internet Relay Chat (IRC). It follows that the history of the individual contributions, along with their associated development decisions, implications and discussions rest stored in the distributed archives of this infrastructure, which captures and tracks the emerging preferences of the emerging FOSS assemblage, while highlighting its limitations.

3. Concluding remarks

In light of the aspects sketched above it is possible to draw some considerations related to both the concept of participation into designing processes, and to the broader idea of designing *per se*.

For what concerns participation as a phenomenon related to a 'better design', there are two aspects to consider. On the one hand, it is no longer confined within the designing phase, as traditionally understood. It extends into the use phase and it becomes an indicator of the validity, success and efficiency of the technological artefact. A well-designed system is one that, not only has few bugs and works efficiently at the level of the technological artefact, but it is one capable to attract and motivate users into active participation, allowing them to contribute in a satisfactory way and keeping them affiliated to the project. Here, participation is both the means of designing usable and meaningful systems and the goal (or outcome) of well-designed technologies. On the other hand, participation brings to the fore an issue of exclusion from and representativeness in design decisions. This issue was the one that ISS designers tried to minimize through traditional participatory approaches. It is true that users' participation in the continuous design of emerging technologies allows system designers and developers to better tune the artefact to real usage practices and users' requests. However, this fine-tuning happens in relation to actual participants only and exclude marginal users[4] and not-yet-users[5].

For what concerns the design process, emergent technologies portray a fundamentally different process from the traditional one. The idea that system requirements can be inscribed into the artefact thanks to analytical and problem-solving logic and that development can be broke down into 'self-containing', linear, and goal-oriented phases is replaced by an emergent process. Designing is no longer confined in a specific time frame, neither in the same spatial space. It implies the continuous, parallel and yet interrelated processes of identifying requirements, implementing changes and evaluating them. Designing is no longer an easily identifiable activity confined within clear boundaries and stated goals. On the contrary, it is a process that needs to be reconstructed by observing how people make sense of what is needed and what is the best way to implement an answer to these needs, by building on the knowledge that is dispersed throughout the projects' infrastructure and amongst the people they collaborate with.

As such, while in traditional ISS development, Winograd and Flores' definition of design could be understood more directly in its substantive terms, in the case of emergent technologies this definition acquires a new meaning. In the former case, designing is the phase between requirements analysis (i.e. understanding) and implementation (i.e. creation), it is the bridge allowing the two

[4] Those who 'only' use the artefact but are not involved in the participatory activities of the project. For instance, they never submit any kind of feedback.

[5] For some reflections on the relevance of non-users in technology production see Wyatt (2003).

phases to interact. In the latter, there is no clear-cut separation amongst phases: requirements analysis and implementation are continuous processes that happen without the formal mediation of a design phase. Therefore, design no longer portrays an interaction. Designing becomes the continuous sensemaking of that enacted and ongoing interaction.

References

Avison, D.E. and Fitzgerald, G. (2003) *Information Systems Development: Methodologies, Techniques, and Tools*, McGraw-Hill Higher Education, 2nd ed.

Beck, K. (2000) *Extreme programming explained. XP Series*, Reading, Mass., Addison-Wesley.

Beynon-Davies, P., Carne, C., Mackay, H. and Tudhope, D. (1999) *Rapid application development (RAD): an empirical review*, in "European Journal of Information Systems", 8 (3), pp. 211-223.

Bijker, W. E. (1995) *Of bicycles, bakelites, and bulbs: Toward a theory of sociotechnical change*, Cambridge, Mass., MIT Press.

Bijker, W.E. and Law, J. (eds) (1994) *Shaping Technology/Building Society. Studies in Sociotechnical change*, Cambridge, Mass., MIT Press.

Bowker, G.C. and Star, S.L. (2000) *Sorting Things Out: Classification and its Consequences*, Cambridge, Mass., The MIT Press.

Bødker, K., Kensing, F. and Simonsen, J. (2004) *Participatory IT Design: Designing for Business and Workplace Realities*, Cambridge, Mass., MIT Press.

Brooks, F.P. (1997) *No silver bullet: Essence and accidents of software engineering*, in "IEEE computer", 20, pp. 10–19.

Bruns, A. (2008) *Blogs, Wikipedia, Second Life, and Beyond: From Production to Produsage*, New York, Peter Lang Publishing.

Dittrich, Y., Eriksén, S. and Hansson, C. (2002) *PD in the Wild: Evolving practices of Design in Use*, Participatory Design Conference, Malmö, Sweden, CPSR, pp. 124–134.

Ehn, P. (2008) *Participation in design things*, In "Proceedings of the Tenth Anniversary Conference on Participatory Design 2008", pp. 92–101.

Farshchian, B.A., and Divitini, M. (1999) *Using email and WWW in a distributed participatory design project*, in "ACM SIGGROUP Bulletin", 20, pp. 10–15.

Gasser, G. R. L., Scacchi, W. and Penne, B. (2003) *Understanding continuous design in F/OSS projects*, in "16th International Conference Software & Systems Engineering and their Applications."

Goldman, R. and Gabriel, R.P. (2005) *Innovation happens elsewhere – Open source as business strategy*, San Francisco, CA, Morgan Kaufman Publishers.

Greenbaum, J. (1993) *PD a personal statement*, in "Communications of the ACM", 36 (6), pp. 47.

Greenbaum, J.M. and Kyng, M. (1991) *Design at Work: Cooperative Design of Computer Systems*, Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Gumm, D.C. (2006) *Distributed participatory design: An inherent paradoxon?,* Denmark, Helsingoer.

Henderson, A. and Kyng, M. (1991) *There's no place like home: Continuing design in use*, in *Design at work: cooperative design of computer systems*, Hillsdale, NJ, L. Erlbaum Associates Inc., pp. 219-240.

Iivari, N. (2009) *"Constructing the users" in open source software development – an interpretive case study of user participation*, in "Information Technology & People", 22 (2), pp. 132–156.

Jones, J. (1983) *Continuous design and redesign*, in "Design Studies", 4 (1), pp. 53-60.

Kensing, F. and Blomberg, J. (1998) *Participatory design: Issues and concerns*, IN in "Computer Supported Cooperative Work: The Journal of Collaborative Computing", 7 (3/4), pp. 167-185.

Larman, C. and Basili, V.R. (2003) *Iterative and incremental development: A brief history*, in "Computer", 36 (6), pp. 47-56.

Latour, B. (1987) *Science in Action: How to Follow Engineers and Scientists Through Society*, Cambridge, Mass., Harvard University Press.

MacKenzie, D.A. and Wajcman, J. (eds) (1985) *The Social Shaping of Technology: How the Refrigerator Got its Hum*, Milton Keynes, PA, Open University Press.

Martin, D., Rooksby, J. and Rouncefield, M. (2007) *Users as contextual features of software product development and testing*, In "Proceedings of the 2007 international ACM conference on Supporting group work", Sanibel Island, Florida, ACM, pp. 301-310.

Michlmayr, M., Hunt, F. and Probert, D. (2007) *Release management in free software projects: Practices and problems*, In "Open Source Development, Adoption and Innovation", Limerick, Ireland, pp. 295-300.

Napoli, P.M. (2010) *Audience Evolution: New technologies and the Transformation of Media Audiences*, New York, Columbia University Press.

Oudshoorn, N. and Pinch, T.J. (Eds.) (2003) *How Users Matter – The Co-Construction of Users and Technology*, Cambridge, Mass., MIT Press.

Raymond, E.S. (1999) *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Cambridge, Mass., O'Reilly Media.

Royce, W.W. (1970) *Managing the development of large software systems*, in "Proceedings of IEEE Wescon", 26, p. 9.

Schuler, D. and Namioka, A. (eds) (1993) *Participatory design: Principles and practices*, Hillsdale, NJ, CRC Press.

Sommerville, I. (1995) *Software Engineering*, International computer science series, Wokingham, England, Addison-Wesley, 5th ed.

Star, S.L. (1999) *The ethnography of infrastructure*, in "American Behavioral Scientist", 43 (3), pp. 377–391.

Suchman, L. (2002) *Located accountabilities in technology production*, in "Scandinavian Journal of Information Systems", 14 (2), pp. 91-106.

Tonkinwise, C. (2003) *Interminable design: techné and time in the design of sustainable service systems*, in *Techné's strategic nature*, Barcelona, European Academy of Design, pp. 1-16.

Winograd, T. and Flores, F. (1986) *Understanding computers and cognition: a new foundation for design*, Noorwood, NJ, Ablex Publishing Corp.

Wyatt, S. (2003) *Non-Users also Matter: The Construction of Users and Non-users of the Internet*, In N. Oudshoorn and T.J. Pinch (eds), *How Users Matter – The Co-Construction of Users and Technology*, Cambridge, Mass., MIT Press, pp. 67-79.

Giacomo Poderi Università degli Studi di Trento
Dipartimento di Sociologia e Ricerca Sociale
Via Verdi, 26, 38122 - Trento
Email: giacomo.poderi@soc.unitn.it