Designing with socio-technical aspects in mind starts with University courses: an experience within an HCI course

Laura Tarantino

University of L'Aquila, Via Vetoio, L'Aquila, I-67100, Italy

Abstract

The socio-technical approach to system design puts under a magnifying lens human, social, and organizational factors of a system underlining that it cannot be technology alone to guide design processes. Though the approach has the potential of influencing IT design, nowadays IT engineers' and computer scientists' views have greater impacts on the shape of IT products than those of ST researchers, also due to the current pace of technology innovation, which makes not mature IT products more and more diffuse. To mitigate the risk of technology-driven products, it is proper to ensure that such views embed human, social, and organizational factors as a second nature, starting from technology-oriented University curricula. This paper reports on a teaching experience in an "Interactive Systems Design" course within an Engineering program, by discussing the project-based learning path proposed to students and in particular addressing ingredients that allow to overcome skepticism and mistrust that students of technological programs usually have towards less technical issues..

Keywords 1

Socio-technical design, user-centered design, design thinking, impact of teaching methods

1. Introduction

The Socio-Technical System (STS) approach to design – evolved from work conducted at the Tavistock Institute [15] – is based on open systems theory emphasizing the fit between social and technical systems and the environment (e.g., [1,3,4,5,15]). It put under a magnifying lens human, social, and organizational factors of a system underlining that it is not technology alone to guide design processes. Originated in the social science realm, socio-technical tools and approaches spread beyond it towards the IT field [9], with the potential of influencing IT design and a consequent wider impact than before, provided that socio-technical thinking becomes accepted within the design orthodoxy of IT professionals [6]. Given IT pervasiveness, Clegg observes that new technologies "*offer opportunities to work in more interconnected ways, providing scope and catalyst for new working arrangements"* [5]. The COVID-19 emergency has clearly imposed a new accelerated pace to this phenomenon, changing our relationships in almost all of our spheres (personal, social, and working) and moving a variety of ICT tools and applications from discretionary to non-discretionary use: the 'social' and the 'technical' have never been so interdependent as nowadays, making a correct socio-technical perspective in system design maybe more important than ever.

It is indeed widely recognized that techno-centric approaches to system design that do not properly consider the relationships between the organization, the people enacting business processes, and the systems supporting these processes may cause failures and lead to systems that do not meet the expectations [1,5,17], implying also a business risk for products may not be chosen by customers. Anyhow, this notwithstanding, socio-technical approaches are still underutilized. Several researchers maintain that the reason for this insufficient consideration of STS concepts and methods is to be searched in its rather philosophical vision, with an overemphasis on the social system and a not sufficient emphasis on the design of the technical system [22]: though different sets of socio-technical

Proccedings of 6th International Workshop on Socio-Technical Perspective in IS Development (STPIS'20), June 8–9, 2020
EMAIL: laura.tarantino@univaq.it

design principles has been proposed [3,4,5], as [1] observes, "*STS design methods mostly provide advice for sympathetic systems designers rather than detailed notations and a process that should be followed*", remaining not appealing enough for technical professionals.

Contaminations with other fields sharing the open-system view and the attention for human and social aspects but working from a more technical oriented perspective (like Information Systems, Computer Supported Cooperative Work, Human-Computer Interaction) may be beneficial for a convergence towards principles, methods, and ultimately a practical vision more connected with technical engineering issues and then appreciated also by IT professionals. The HCI community, in particular, is clearly a good ally in this direction, for its explicit consideration of human and social aspects, and its repertoire of User Centered Design (UCD) methods and techniques oriented to the analysis of users, contexts of use, working practices, and organization structures, as well as to design organization, systems specifications, and evaluation (e.g., [2,8,14,16,18]). Furthermore, HCI appears as complementary to STS design as to the interaction between people and technology, somehow under considered by STS approaches. Anyhow, notwithstanding the beneficial potential of such cross-contamination, the path towards technological innovation balancing the three legs of human-centered product development (user experience, marketing, and technology) is not to be taken for granted without some action. Unbalanced IT products are made more and more frequent by – among others – the current pace of technology innovation, the diffusion of agile development approaches, and the ease of products' delivery (e.g., through application stores), which make IT products more and more dependent on the views of IT engineers, computer scientists, and even technology enthusiasts early adopters: more and more often we interact with novel IT products still in the area of "unfilled need" of the need-satisfaction curve (see Figure 1).

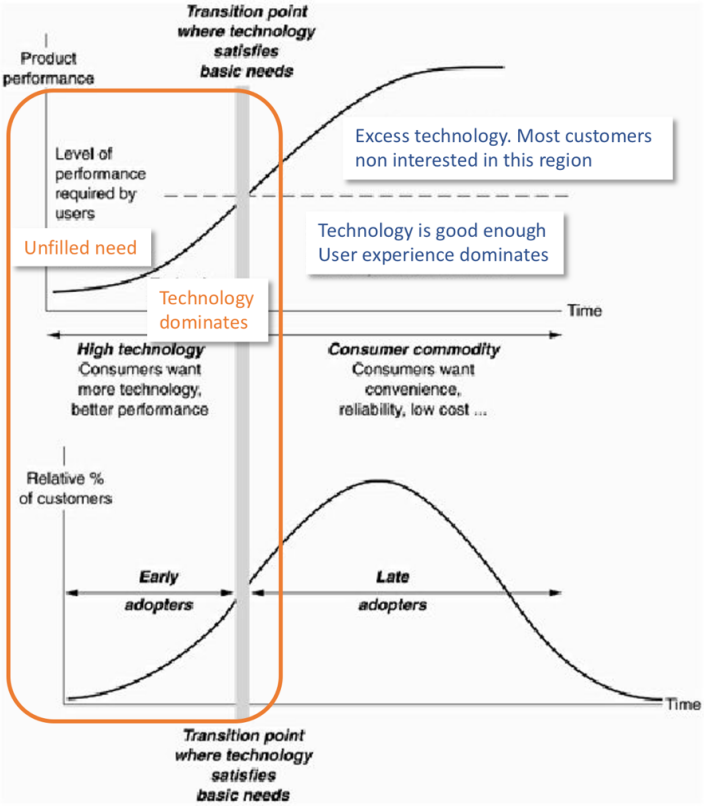


Figure 1: The need-satisfaction curve of a technology compared with the change in customers as a technology mature: in the early phases, innovators, visionaries and technology enthusiasts drive the market, though they are only a small percentage of the market and do not represent the view and the needs of the majority of the users (modified from [18]).

Given this situation, we agree with [1] maintaining that it is not enough to simply analyze a situation from a socio-technical or a human-centered perspective and then explain the analysis to engineers. It is necessary a gradual introduction of socio-technical and human-centered considerations into existing SW procurements and development process with a mindset shift on the engineering and design side. Our view is that, while sensitization and awareness activities advocated by [1] keep going on with existing stakeholders, a new generation of engineers is to be educated, by presenting them socio-technical thinking in technical university courses as an essential design component, to ensure that developers' views embed human, social, and organizational factors as a second nature, to be learned and digested from the beginning, intertwined with technical design principles.

This paper gives a contribute in this direction, by discussing our teaching experience in the *Interactive Systems Design* course within the *Computer and Systems Engineering* Master at the University of L'Aquila. In particular the paper discusses the project-based learning path proposed to students, addressing in particular the ingredients allowing to overcome the skepticism and the diffidence that students of technology courses typically have towards anything "not nerd enough". The final objective is to make them acquire a design attitude and a designer view integrating human, social, and organizational factors as essential and inescapable features.

The remainder of the paper is structured as follows: after briefly surveying in Section 2 obstacles typically encountered by teachers of less technical courses, Section 3 discusses how they are faced by the course under analysis to make a methodological course appealing to engineering students, and finally, in Section 4, conclusions are drawn

2. Obstacles and enemies

Shortly after his book "The Human Interface" [20] was published, Jef Raskin published on his official website a note received from a colleague who experienced difficulties in explaining Raskin's interface concepts to people. The note said that the situation was somehow similar to the scene from "The Matrix" movie where Morpheus explains Neo who the agents are and so he made a parody of the scene dialogue to describe the situation [2]: "*The current interfaces are a system, Neo. That system is our enemy. But when you're using these interfaces, you look around, what do out see? Icons, file names, modes, hidden interface elements, and people attempting to use them, the very minds of people we're trying to save. But until we do, these people are still a part of that system, and that make them our enemy. Most of them are not ready to change interface, and many of them as so hopelessly dependent on the system, that they will fight to protect them.*" [23]. Nowadays the way we look at (and design) interactive systems has considerably broadened, including issues going well beyond user interface aspects, but the problem has remained more or less the same: a strong resistance to change and a strong opposition to concepts and methodological tools that lay outside the technological stream, coming not only from systems designers but also from students enrolled in technological programs, skeptical and mistrustful of "not technical enough" stuff.

This seems to be a common problem reported by many colleagues, as testified for example by an online discussion of few months ago, involving some teachers of HCI courses in different Italian universities within degree programs in Computer Science and Information Engineering. The discussion stemmed from a post where one of us quoted with some bewilderment opposite outcomes from student evaluation questionnaires: while some students were very positive about the course program, content and teaching method, others labeled the course as boring and with no important content. One colleague underlined that students' lack of appreciation is to be found in the course objectives: "*In my experience, Computer Science students who belong to the nerd type dislike anything not nerdy enough, like HCI or requirements. It's not related to course quality nor teaching skills"*, while another observed that opposite evaluations are not rare: *"I always have bimodal evaluation like – dislike"*. Another observed that the teacher has no choice but to follow committee decisions on course objectives and content: *"The student says that course contents are bad. This is not a teacher's choice; it is mostly an issue agreed upon the entire degree committee. This should be made clear to students"*.

[2] The whole note from Danny Lewis was originally at human.sourceforge.net/the/matrix.html. The page is no longer accessible but the citation of the Matrix parody can be still found at https://instant-thinking.de/2004/03/08/interface-design/

In our opinion, the latter position is questionable, since, while overall course objectives and content are actually agreed with committees, the teaching methods and the proposed learning path are under the teacher's control and can be used to conceive a course appealing to students and, with the right ingredients, acting as a "trojan horse" for HCI/STS points of view in engineering studies. Going back to the Matrix parody, if students fight to protect their own system centered view (and they will do it), we have to fight back. In the following section we discuss how the "fight" against obstacles and enemies is conducted within the course under analysis.

3. Obstacles and enemies

The *Interactive Systems Design* course of the University of L'Aquila is offered within a *Computer and Systems Engineering* Master to students with background in Information Engineering. Most students come from a Bachelor program of the University of L'Aquila where, beside Computer Science courses (on programming, computer architectures, operating systems, databases, web applications, computer networks) they attend introductory courses on Electronics, Automation, Telecommunication, and Electrical Engineering. After three years spent addressing the system from a purely technical point of view and looking for solutions to clearly defined given problems, the *Interactive Systems Design* course demands a complete shift in the student thinking approach, requiring them to define problems (before solving them), to look at the system from an holistic point of view, and, ultimately, to make the transition from "problem solvers" to "designers".

3.1. Overall course objectives and approach

The course aims to provide the knowledge necessary to design usable interactive applications, with a strong methodological approach. Based on project-based learning, the course incrementally leads students through the phases of field study and requirements analysis, conceptual design, scenario-based design, paper prototyping, mockup prototyping, usability analysis and evaluation. The overall approach is based on a contamination among HCI, Information Systems, and Socio-Technical Systems views.

As to socio-technical issues, an exhaustive formal presentation of the field is beyond the scope of the course: it would be not only unrealistic in an introductory course to interactive systems design, but also counterproductive, given the background of the enrolled students and their recognized resistance to less technical topics. Rather, socio-technical issues implicitly pervade all projects assigned to students, so that they can perceive and *experience* the necessity of addressing human, social, and organizational issues as an unavoidable component of the design, which hence hopefully becomes a second nature in their work as designers (in particular, the course approach is coherent with Clegg's meta-principles, content principles, and process principles of STSs [5]). Students' teams are assigned a variety of projects with different goals, different contexts of use, and different target users, regularly discussed in plenary in-class presentations, to allow students to be confronted with the diverse problems/issues that a designer has to face. The final exam is a presentation aimed at critically analyzing the work done, from a methodological point of view, in a metacognitive way, addressing the various aspects of the design.

In the following we analyze the course from two perspectives: the methodological tools proposed to work on their thinking approach and the practical project-based path.

3.2. Work on their very minds

Students are not yet designers, which is a double edged starting point: if this implies that we cannot expect from them an immediate adherence to design principles and that we have to debunk the previously acquired system-centered view, on the other hand this allows we teachers to present design principles and approaches from scratch, and, in particular, to present user-centered and socio-technical thinking not as something to be added to a previous view, but as essential components of the designer work.

Step zero: make students appreciate the change (alias the "but we've always done this way" battle)

In her socio-technical design history [15], Mumford starts the discussion by underlying the crucial liaison between *technology and change* and, talking about action research, maintains that "*it will be difficult to use successfully if the parties involved are hostile to each other, disinterested in developing strategy or unwilling or unable to cooperate"*. This kind of hostility is more or less the same that teachers often have to experience with students of engineering courses who look at themselves as "computer masters" with no other perspective of systems than their own. The very basic step is therefore to make them appreciate the crucial role of *change* in technology innovation. The course hence begins with a survey of the evolution of interactive systems and underlying technology, showing evolutionary and revolutionary steps from teletypes to Virtual Reality, and showing how without changes in system views and with computer professional of the sixties stuck to the technology in use then, we would be still using teletype instead of, e.g., fancy augmented reality apps on our smartphones. Insights are given to selected specific innovators' contributions, illustrating also what often remains behind the scene: motivations and inspirations of their creators/designers, along with their successful personal stories (e.g., Vannebar Bush, Douglas Engelbart, Jef Raskin, Steve Jobs are presented as inspiring examples to students). The "but we've always done this way" battle is usually won this way.

Help students keep on track: give them methodological helms and ensure a rigorous approach to design (alias the "common sense" battle)

Students are not designers: not only they have no idea on how to swim in "design waters" but the dangerous myths of "common sense design" and "intuitive design" are always behind the corner when talking about user interface design, user experience, and interactive application design, with a high risk of a not rigorous approach to design. Design frameworks are adequate weapons to win this battle, since they allow students to clearly visualizes *where* they are in their design path, *when* doing what, and *what* to focus their attention on. In this direction, the Hevner's framework [13] on the one side and Design Thinking (DT) [2] and UCD [2,14,19] on the other side may have complementary roles in showing the *what*, the *where*, and the *when* (see Figures 2 and 3).

We observe that the Hevner's framework in this case has to be simplified by focusing just on artifacts and by replacing the Rigor Cycle with a "*rigor recommendation*" (Figure 2), to become a very effective guide forcing students to find sound justifications for each single design choice; though it would not be the best methodological choice for interactive applications falling within the third paradigm of HCI (looking at the interaction as phenomenologically situated [12]), we find it appropriate to propose it as a first framework in an introductory course to HCI that starts by addressing interactive applications falling within the second paradigm of HCI with a focus on information communication, task modeling and users modeling. Anyhow, it cannot work by its own, since, while it helps in clearly identifying *what* to focus attention on, it does not provide guidelines about *when* doing *what*. On the other side, Design Thinking (Figure 3-(a)) is really effective in helping students in the transition from a modus operandi in which they are given predefined problem to solve and a modus operandi in which they have to define *what is the right thing to do* (problem definition) before finding *how to do the thing right* (problem solution).

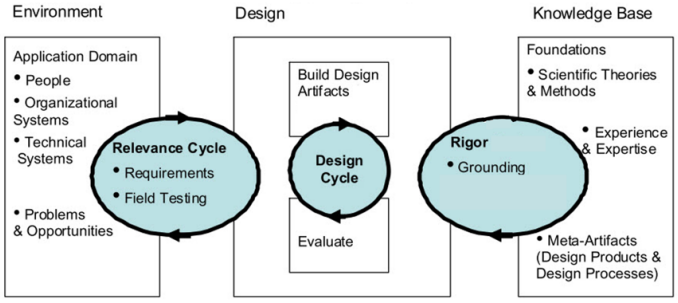
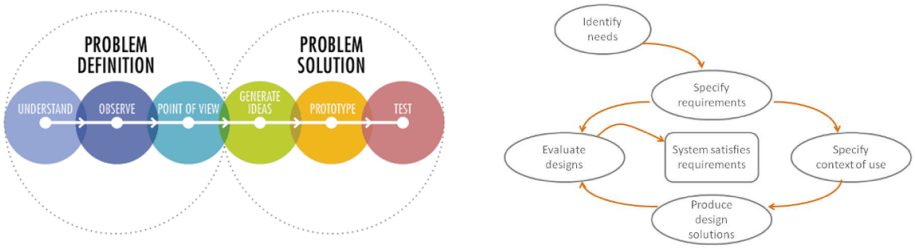


Figure 2: A reference design framework derived from the Hevner's proposal in [13],



(a) (b)

Figure 3: Reference design frameworks: (a) the Design Thinking framework, and (b) the User Centered Design life cycle

Make students diverge as soon as possible (alias the "we have the one solution right away" battle)

Engineering Master students come from learning experiences strongly characterized by the search of *the* solution (sometimes a single number!), starting from the problem data and by means of sound logical reasoning, whereas design is a territory of alternatives, choices, dismissions, trade-off, discard. Clegg's meta-principles 3 and 7 make it very clear [5]: "*Design involves making choices*" (e.g., on how the overall system will operate and on what form of technology will be required), often non deterministic and leaving a certain degree of freedom, and, since "*Design is contingent*", there is no "one best way" and solutions do not have universal applicability. Students have to learn to consider alternatives, comparing them, and discarding some of them on the basis of selected criteria. In our experience this is one the most difficult battle to win: Engineering students tend to point straight to "the one solution", sometimes ignoring altogether the "problem definition" DT macro phase. The DT framework proves to be beneficial in this aspect too, underling in a clear way the alternation of divergent (creating choices) and convergent (making choices) thinking (see Figure 4). Before the explicit introduction of the Design Thinking approach in the course, it was not so rare to see students focusing on one alternative only, falling in love with it, and getting so attached to it to the point of crying if forced to discard it as a result of evaluation. The sooner students learn to diverge, the sooner they will accept "alternative discard" as a regular design ingredient and not as a design failure.

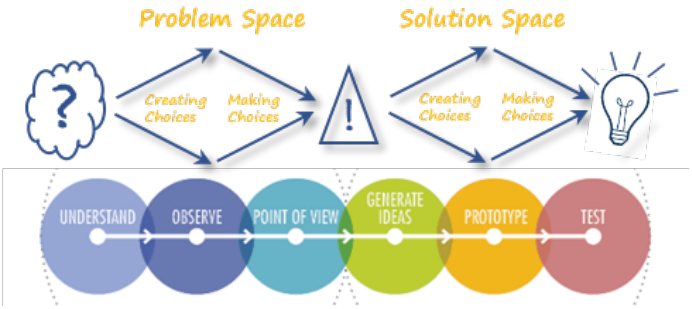


Figure 4: Alternation of divergent and convergent thinking

Make students think as designers: start from the end (alias the "this is useless boring stuff" battle)

As designers know, there are no crisp boundaries between the DT phases or the UCD phases, not only because iterative design makes us take steps back to modify previous choices, but also because the experienced designer may use also forward-thinking and starts considering from the beginning some design aspects formally belonging to future phases. As a consequence, even if apparently paradoxical, within a project-based course it is risky to present the different topics, methods and techniques by strictly following the relative order in which they are usually utilized during the design process, since

this choice would prevent a controlled and sound forward thinking. Our approach is to start from a general overview of the global picture and of the main aspects of the different phases, followed by insights of specific methods and techniques starting from issues pertaining the prototype phase (like visual design and interaction design). This choice has a number of advantages. First of all, it allows the teacher to capture engineering students' attention from the beginning with topics closer to their interest and less boring for them than, e.g., methods for user requirement analysis would be. Furthermore, if paralleled with some (possibly "quick-and-dirty") design-and-action theory prescribing "how-to-do-something", this make them operative from early phases of the course and able to develop some simple artifact, thus making it possible a teaching approach based on a sequence of projects with increasing difficulty and increasing formal quality, during which students themselves will recognize the necessity of (and will ask for) methods and techniques otherwise judged boring.

3.3. The practical design path

The basic point of STSs is that "Design is systemic", which, in his recommendations, Clegg comments as follows: "*A sociotechnical perspective explicitly embraces the idea that all aspects of a system are interconnected, that none should take logical precedence over the other, and that they should be designed jointly. Technical and social systems are interdependent. Exclusive emphasis on any one component during design, for example on technology, will be sub-optimal.*" [5]. Anyhow, though such "aspects compresence" is surely beneficial for the design, we believe that things are a little bit different when talking about teaching approaches, where addressing separately the different issues (before merging them into a unifying vision) allows students to better reflect on each facet of the system and of the design process, and to appreciate its contribution to the overall picture. Furthermore, the demand of a complete systemic view since the beginning might make the design learning difficult and discourage students.

Following this line of reasoning, typical levels of an interactive system [24] (mechanical, informational, personal, community, as in Figure 5) are mirrored by project ingredients/requests, so that students (1) realize that such levels are ways to view the system and not ways to partition it, and (2) are forced to think according to each view and to interdependencies among different views, within a path of four projects with increasing difficulty, within which they are requested to: adapt the product to a *specific device*, model structured/unstructured *(heterogeneous) information* and design access structures to it, address universal/individual *psychological aspects*, model *users*, model *tasks*, address *context of uses*, address *organizational rules/constraints*, address *social interaction*, model *community experiences.*

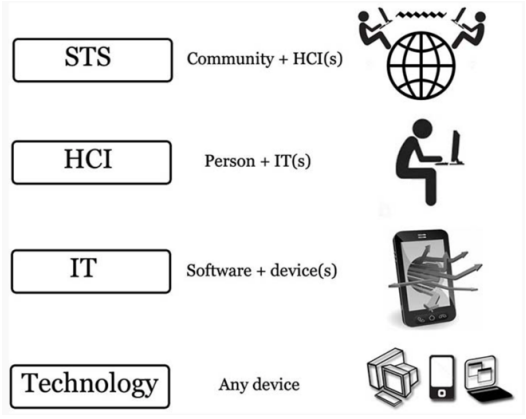


Figure 5: Interactive system levels (from [24], unknown author)

All projects are driven by Design Thinking and UCD with the common final aim of designing, realizing, and evaluating interactive mockup prototypes. Table 1 summarizes main learning objectives of the four project assignments (projects of level L_i rely on learning achievements of projects of level $L1$ to $Li\text{-}1$). The first three levels are focused on users/contexts/tasks students are familiar with and act

as a training ground in view of the 'big' exam project of level L4. Notice that, though the general theme is the same for all student teams in levels L1 to L3, assignments of different teams differ in specific requests about the tasks, the type of users, and the context of use. By comparing each other proposed artifact in plenary in-class discussions, students make a first-hand direct experience of the lack of "the one universal solution" in design. As to users' studies, projects of level L2 and L3 address predefined users' types (first time, novice, intermittent, expert/frequent) to start by addressing universal psychological facts. Furthermore, teams are assigned different types in the two levels for experiencing how a change of the users' type impact on the design. Finally, the project of level L4 makes a step forward from generality to specificity requiring a "true" users study to address differences across individuals and groups with respect to the expected usage of the system. As to social and organization aspects, they are included in all project of levels L2 to L4; in particular, all L4 projects include some kind of social feature in the system. It is interesting to notice that, though not explicitly requested, students' teams sometimes choose to include system social capabilities also in projects of previous levels.

Table 1
The path of team assignments and objectives

Level	Project focus	Learning objectives
L1	Interacting with a smart TV (common them, different teams address different user's intentions)	Supporting specific user's intentions, articulating intentions, modeling few specific tasks, addressing consistency among tasks, addressing a specific device
L2	Booking a meal at the University canteen via a tablet-based app (common theme, different teams address different users' type)	Formalizing simple users' requirements, addressing a predefined user type (first time, novice, intermittent, expert/frequent), modeling a multi-step procedure, addressing organizational rules/constraints, addressing a specific more constrained device
L3	Designing an innovative context-specific hot/cold drinks vending machine (common theme, different teams address different scenarios, like a rock concert, a laboratory, a train station)	Formalizing users' requirements, addressing a predefined user type (first time, novice, intermittent, expert/frequent) different from the previous one, addressing contexts of use, modeling multi-step tasks, thinking out of the box, addressing organizational rules/constraints, addressing groups of users acting as a whole, making choices about technology
L4	The "big" team project (individual theme)	Using data gathering techniques, observing users, asking users and experts, modeling users, making choices about technology, addressing organizational rules/constraints, addressing social aspects, addressing community purposes and policies

4. Discussion and conclusions

The course structure discussed so far is the results of around 15 years of experience that gradually modified the course syllabus and approach, based on a continuous analysis of achieved results. Starting from a "pure" HCI course mostly based on UCD, the most sensible improvements came from: (1) the shift from the lecture based approach of the early editions, with project developed *after* the course, to the current project-based approach with students working on their projects *during* the course *with the teacher*, (2) the introduction of the Hevner-like design framework as "rigor recommender", and (3) the

introduction of the Design Thinking framework with the explicit reference to the alternation of divergent and convergent thinking.

One of the most critical choice was to decide whether to maintain implementation aspects as part of the student work. Implementation is a real double-edged weapon: if not included, there is a significant risk that Engineering students may perceive the course as "not nerd enough" but, when included, as this course experience showed, often students do not explore all design options in fear of future difficult implementation work. We decided to make the course less and less focused on implementation aspects (anyhow learned in other curricular courses) and more and more focused on methodological ones and opted for a trade-off choice: in L4 projects student teams are requested to conduct feasibility studies and to carefully address the consistency of design choices with the selected technological platform, without implementing the "engine" of the system while realizing a mockup interactive prototype. In summary, to take care of students' skepticism with respect to "less technical subjects" and to carefully single out the ingredients necessary to make the course act as a "trojan horse" for HCI/STS points of view in engineering studies, exam projects are conceived according to the following measures:

- to boost students' interest, exam projects rely on technological infrastructures that provide an ample variety of hints for individual assignments on specific technical aspects (e.g., Recommender Systems, Augmented/Virtual Reality);
- to make them reason on community rules/policy/purposes, exam projects include social aspects;
- to make them perceive the usefulness of considering personal and community levels, exam projects address student problems and communities students belong to.

In conclusion we observe that nowadays bridging the gap between the engineering mindset and the softer design mindset is more and more mutually beneficial, since, on the one hand, technology becomes more and more pervasive thus heavily influencing soft design decisions, while, on the other hand, the technological innovation pace make engineers more and more often face novel problems without ready-made solutions: crafting new skills from the beginning is therefore a must.

5. References

[1] G. Baxter, I. Sommerville, Socio-technical systems: From design methods to systems engineering, Interacting with computers 23 (2011) 4–17.

[2] T. Brown, Change by design: How Design Thinking transforms organizations and inspires innovation, HarperCollins e-books, 2009.

[3] A.B. Cherns, The principles of sociotechnical design, Human Relations 29 (1976) 783–792.

[4] A.B. Cherns, Principles of sociotechnical design revisited, Human Relations 40 (1987) 153–162.

[5] C. Clegg, Sociotechnical principles for system design, Applied Ergonomics 31 (2000) 463–477.

[6] M.C. Davis, R. Challenger, D.N.W. Jayewardene, C.W. Clegg, Advancing socio-technical systems thinking: A call for bravery, Applied Ergonomics 45 (2014), 171–180.

[7] C.S. de Souza, J. Preece, A framework for analyzing and understanding online communities, Interacting with Computers 16 (2004) 579–610.

[8] A. Dix, J. Finlay, G.D. Abowd, R. Beale, Human Computer Interaction, 3rd ed. Addison-Wesley, Harlow, UK, 2004.

[9] K. Eason, Sociotechnical systems theory in the 21st century: Another half- filled glass?, in: D. Graves, D (Ed.), Sense in social science: A collection of essays in honour of Dr. Lisl Klein, 2008, pp. 123–134.

[10] S. Gregor, The nature of theory in Information Systems, MIS Quarterly, 30 (2006) 611–64,.

[11] J. Gulliksen, B. Görannson, I. Boivie, S. Blomkvist, J. Persson, Å. Cajander, Key principles for user-centred system design, Behaviour & Information Technology 22 (2003) 397–409.

[12] S. Harrison, P. Sengers, The Three Paradigms of HCI, in: CHI 2007 Proceedings, 2007.

[13] A. Hevner, A three cycle view of Design Science Research, Scandinavian Journal of Information Systems, 19 (2007) 87–92.

[14] International Standards Organisation: Ergonomics of Human–System Interaction – Part 210: Human-centred Design for Interactive Systems, ISO, Geneva, Switzerland, 2010.

[15] E. Mumford, The story of socio-technical design: reflections in its successes, failures and potential, Information Systems Journal 16 (2006) 317–342.
[16] J. Nielsen, Usability Engineering, Academic Press, London, UK, 1993.
[17] D.A. Norman, Things that make us Smart: Defending human attributes in the age of the machine, Addison-Wesley, Boston, MA, 1993.
[18] D.A. Norman, The invisible computer: why good products can fail, the personal computer is so complex, and information appliances are the solution, MIT Press, 1998.
[19] D.A.Norman, S. Draper (Eds.), User Centred System Design, LEA, Hillsdale, NJ, 1986.
[20] J. Raskin, The Human Interface: new directions for designing Interactive Systems, Addison Wesley, Reading, Massachusetts, 2000.
[21] F. Ricci, L. Rokach, B. Shapira, Recommender Systems Handbook, 2nd ed., Springer US, 2015.
[22] G. Salvendy (ed), Handbook of Human Factors and Ergonomics, 4th ed., John Wiley & Sons, Inc., Hoboken, New Jersey, 2012.
[23] D. Wegner, Interface Design, 2004. URL https://instant-thinking.de/2004/03/08/interface-design/.
[24] B. Whitworth, A. Ahmad, Socio-technical system design, in: C. Stephanidis (ed.), The encyclopedia of human-computer interaction, 2nd ed., Interaction Design Foundation, 2012, chapter 24.