THE SOCIOTECHNICAL APPROACH
THE CHALLENGES FOR SOFTWARE ENGINEERING

Henrique Cukierman[1]

INTRODUCTION

It is necessary to take into account the new social ordering that has been produced since the emergence and adoption of new information and communication technologies in order to build a better quality of life and a fairer society. These new technologies are seen as sources of radical changes: if so, they constitute a scenario which significantly transforms various dimensions of modern life, as the nature and experience of interpersonal relations and communications; the relations and conditions of work; the operation modes of the worlds of business, culture and arts; the formulation of regulatory policies. In summary, the new technologies change the form and substance of social control, participation and cohesion. However, at the same time, they are also modified by the social experience. It is the case of SE (Software Engineering) and, in fact, its academic and professional community usually highlights the importance of social, cultural, political and organisational issues in software development projects and in the improvement of software engineering processes.

However, it is common to see these issues qualified as "non technical", indicating that part of the SE community believes it is possible to split the problems into "technical" and "non-technical" ones, despite the fact that it becomes increasingly clear that such a division cannot help facing the growing challenges of SE. This division leads to the supposition that those problems related to the first class – the "technical" – have greater importance. The second class of problems – the

1 Universidade Federal do Rio de Janeiro Programa de Engenharia de Sistemas e Computação/COPPE – Programa de História das Ciências e das Técnicas e Epistemologia (Engenharia de Computação e Informação/Escola Politécnica. Avenida Horácio Macedo 2030, Centro de Tecnologia, Bloco H, sala 319. Cidade Universitária, Rio de Janeiro, RJ,21941-914, Brazil). Email: hcukier@cos.ufrj.br.

"non-technical" – do not receive any positive qualification. Instead they are defined by exclusion, by the negative: "**non** technical".

Granted with a relatively smaller concern, the social, cultural, political and organisational issues are relegated to disciplines external to SE. The development of software systems seems wrapped in a "technical orthodoxy", and therefore is seen as a "technical" process, to be carried out only by its specialists. However, the effort required for the development of these systems presents problems and challenges of complexity far beyond "pure" software techniques, which requires the intervention of different kinds of knowledge coming from other disciplines. For this reason, the SE community should not avoid being "contaminated" by the contributions of human and social sciences. Fortunately, many researchers have noticed that information technologies are also inevitably social, and therefore have been trying to focus on SE as a problem of social complexity. HERBSLEB [2, pp. 24] proposes that:

> […] understanding how to do software engineering better requires a deepening of our understanding of 1) effective software engineering principles and practice, and 2) how these principles and practices line up against human cognitive, social, and cultural functioning. Current software engineering research, in my view, is making steady progress in the former, but constantly risks irrelevance as it neglects the realities of the latter.

The way out of the impasse created by the separation between the technical and the social demands a change in the angle of approach. It is a matter of using a new framework, in which the technical and the social are constituted through a movement of co-modification, only captured by a concomitantly technical and social approach, by a **sociotechnical approach**. An approach that seeks to face the "seamless cloth" which imbricates in SE the technical and the social in the same and indivisible fabric.

THE CHALLENGES OF THE SOCIOTECHNICAL APPROACH

Imbrication, indissociability and indetermination of the social and the technical underlie the sociotechnical approach, which conceives them as a mutual determination, and therefore only treatable by an interdisciplinary (or even transdisciplinary) approach.

BOEHM [3, p.12], based on a dictionary definition of SE – *"the application of science and mathematics by which the properties of software are made useful to people"* – proposes on the subject of the kinds of knowledge involved in SE that *"'useful to people' implies that the relevant sciences include the behavioral sciences, management sciences, and economics, as well as computer science"*. HERBSLEB [2, p.25] sheds more light on the disciplinary limits of SE:

> [...] product architecture and organizational structure are intimately related [...] If we assume that we can design architectures purely on technical grounds, we place our organizations and our customers at risk, but as yet we understand relatively little about how to think about this problem, [...] We need interdisciplinary research to understand the constraints that architectures impose on organizations, and that organizations impose on architectures, and how technical and organizational structures can co-evolve.

With this definition, Herbsleb offers a very interesting version of the sociotechnical approach. An approach accurately perceived by Tracy Kidder in his book *The soul of a new machine* (apud LATOUR [4]) about the history of the construction of the Data General Eclipse throughout the 1970s. A history that begins with Tom West, head of the development team of a computer internally still called Eagle, a 32-bit machine built to compete with the 32-bit VAX from DEC (Digital Equipment Co.). In order to have an idea about the competitor's computer, West arranged a secret visit to a company that owned the VAX then recently released:

> Looking into the VAX, West had imagined he saw a diagram of DEC's corporate organization. He felt that VAX was too complicated. He did not like, for instance, the system by which various parts of the machine communicated with each other, for his taste, there was too much protocol involved. He decided that VAX embodied flaws in DEC's corporate organization. The machine expressed that phenomenally successful company's cautious, bureaucratic style. [4, p.5]

Here we have a perfect example of the sociotechnical approach, which frames the technical and the social in a same view, a synoptic view, as the one that animates the description of Tracy Kidder, but also James Herbsleb's and Bruno Latour's arguments. In his *Science in action* [4, p.5], Latour calls attention to a fundamental attribute of the sociotechnical approach, namely: "Context and contents merge".

So, how to navigate through such complexity? SUCHMAN [5], in her *Plans and situated actions*, offers valuable answers for the differences between plans and situated actions, based on the research of the anthropologist Thomas Gladwin, published in 1964, about fishermen in Micronesia. He drew attention to the navigational techniques of one of those people, the Trukese, used for long trips on sea, pointing to the contrasts between their way to navigate and that of the Europeans. For the purpose of this article, that of discussing SE practices, it is reasonable to relate the concept of plan to that of model, and it is for this reason that, by referring the issues of Lucy Suchman and Thomas Gladwin, we use both notions as synonymous, bringing them all together under the denomination "plan/model".

According to Gladwin, the European navigator begins with a plan/model – a course – designed according to certain universal principles, to which he relates all the movements of his trip, and, therefore, his effort is to keep the course as previously planned. If unexpected events occur, the European navigator has first to change the plan/model and only after reacts properly. The Micronesian navigator starts with a goal instead of a plan/model, leaving towards his goal and responding to contingencies as they appear in an *ad hoc* style. Thus, he makes full use of the information provided by the wind, the waves, the tide, the stream, the fauna, the stars, the clouds, the sound of the water hitting the boat, sailing in complete compliance with all of them. His effort is targeted to reach his goal, and if it is easy for him to answer about his goal, nevertheless he can not offer an answer with regards to his course. This effort to pursue a specific goal through an *ad hoc* style is what Suchman calls a *situated action*.

For SUCHMAN [5, p. ix], plans/models are a weak resource in face of *ad hoc* activities. In fact, given the European bias of our culture, we call for a plan/model as a guide only when pressed to account for the rationality of our actions. Previously proposed, plans/models are necessarily vague, in that they must accommodate the unpredictable contingencies of an always particular situation. Rebuilt in retrospect, the plan/model filters the specificities of the details that characterize the situated action in favour only of those actions that can be framed for their eventual conformity to the plan/model.

According to the planner/shaping vision, plans and models are prerequisites for action, prescribing it in detail. However, the course of action can only be designed or rebuilt in accordance with previous intentions and typical situations ("best practices") since the prescriptive meaning of intentions vis-à-vis the situated action are inherently vague. The coherence of situated action is linked not to individual predispositions or conventional rules, but rather to local and contingent interactions in relation to the actors' particular circumstances.

In short, every course of action depends on its material and social circumstances. Instead of abstracting action from its contingencies, representing it as a rational and universal plan/model, the proposed approach is to investigate how

those involved with action can use their circumstances to achieve what can be called an *intelligent action*.

SUMMARIZING SOME APPROXIMATE CHARACTERISTICS OF THE SOCIOTECHNICAL APPROACH

If software engineers do not look like fishermen from Micronesia, it is not difficult, however, to understand from such diverse experience the challenges these engineers have considering what is clearly a problem of knowledge construction. A certain fetishization of models on the part of these engineers reverses, in practice, the methodology proposed by Suchman – that of the primacy of situated action over plans/models. Therefore, if a sociotechnical approach to SE is considered, it will have to give up the obsessions with universalizing models in order to accomplish the much more challenging task of addressing the irreducibly local specificities of all situated action. To account for such particularities, or to put it another way, in order to relate defined models to local contingencies, an approach is needed that reaches beyond the mathematical, algorithmic and simplifying reductions undertaken by models. An approach that takes into account the indissociable imbrications of the seamless cloth that performs the real world. A sociotechnical approach, which reaches synotically the technical and the social, must necessarily be an interdisciplinary approach.

Briefly summarizing and making use of dualistic schemas (necessarily simplistic, but of reasonable communicative efficiency), it can be said that for the sociotechnical approach the characteristics described in the subsequent items of this section prevail.

The local, the situated (resisting the global, the universal), the case by case, the contingencies

BOEHM [3, p.25] distinguishes the concern about the local as an obligatory counterpoint to the modern legacy:

> […] the theory underlying software process models needs to evolve from purely reductionist "modern" world views (universal, general, timeless, written) to a synthesis of these and situational "postmodern" world views (particular, local, timely, oral).

The modernist insistence on models capable of apprehending the real in its complexity can only subsist under the hypothesis of a stable and regular world, if not a world made of deterministic causalities. In this regard, TEIXEIRA [6] observes:

> Let's look at information modeling, a key tool with a very strong presence in systems development cycles. It denotes a "naive realistic position", which presupposes that there is only one suitable method for the real, objective world to be discovered. Moreover, if discovered through this method, such a world would be consistent, of manageable complexity, and therefore controllable. The underlying assumption is that there is a priori an ordered world, which means that it is enough to the software engineer to discover/capture the pre-existing requirements, then to formalize a specification and develop the desired system departing from that specification. Most approaches tend to consider that it is possible to define the requirements in advance and that they will remain stable throughout development. [...] With representation in place, development processes assume a hierarchical structure with a strong belief in the power of the designer to have all control of the situation [7]. They leave any consideration about the real world and begin to focus only on the representation of the system [8].

Barry Boehm, based on his research, together with M. Phongpaibul, on the implementation of software improvement processes in Thailand after the adoption of the CMM (Capability Maturity Model) in that country [9], shows that the challenge is not to deploy a universal model, but mainly to project the relationships, roles, and skills that are expected from developers. Although the elaboration of the relations between the universal - present in models, patterns and methods - and the local - the practical use of these models, patterns and methods [7, p.135] - is a typical matter of design, models and patterns usually focus the problem on a simplistic technicalist bias, neglecting cultural, social, and political issues. For this reason, when the implantation of a model is not successful, these issues frequently appear as an explanation for the failure, maintaining the model's aura of "technical" infallibility, thus preserving its presumed competence. But by itself, no model or pattern can guarantee the repetition of a supposed success obtained in some previous situation; in fact, it only replicates itself although claiming the replication of a supposed "intrinsic" competence. In order to be actually used, a universal model or standard will ultimately have to elaborate its meaning locally.

Complexity (instead of simplifications)

Nowadays, the conceptual premise that interprets Information Systems (IS) as sociotechnical systems is increasingly familiar [10]. Therefore, recognising the complexity (development, maintenance or deployment of a process), from its conception to its implementation, should be seen as managing not only the development of code and the infrastructure of networks and computers, but also, for example, the interests of various groups and individuals [11]. Thus, we have what could be called a sociotechnical negotiation process, in which the work of the software engineer, in practice, is necessarily to build sociotechnical networks as a result from his/her negotiations of the roles of indissociable "social" and "technical" entities.

However, historically, software projects have been seen only as the task of specifying (and following) technical standards, leaving aside the complex, and sometimes uncontrollable, task of aligning innovations, technologies, culture, policies, market and organizational conditions [7, pp. 8-9]. It is common the idea, based on a realistic posture, that there is an ordered external world, and consequently that it is possible for the software engineer to discover/capture the requirements that are somehow given in advance. Next, formalized specifications at various levels of abstraction serve as a hierarchical guide to all efforts in constructing software systems, under a comforting and traditional premise, that the designer holds *a priori* total control of the process. The sociotechnical approach, by not dividing *a priori* the complexity of SE into "technical and non-technical aspects", or human and nonhuman aspects, recognizes the exposure of the software designer/engineer to the contingent, the local, the situated. Assuming this approach, we are instigated to detail the perception and description of the concrete mechanisms that make possible the cohesion of the sociotechnical networks that guarantee the existence and success of software products and software processes. It is an approach where models and specifications are entities, like several others that must be conveniently entangled in the sociotechnical network to be stabilized.

Non formalized knowledge

Scientific practice is not only about following universal rules. It consists more specifically of particular courses of action, completely understandable only in their local context (depending even on the people who are involved in such practices, as there is always a personal level of achievement). Universality and independence of context cannot be taken for granted but must be analysed as precarious achievements – for example, as a result of the successful expansion of networks connecting humans and non-humans.

One version of this rejection of universalism can be given by the contrast between explicit knowledge and tacit knowledge. The former is made up of information or instructions that can be formulated into words or symbols and can therefore be stored, copied and transferred through impersonal means such as documents or computer files. Tacit knowledge, in turn, is knowledge that was not (and could not be) formulated completely and explicitly and therefore cannot be effectively stored and transmitted entirely by impersonal means. Motor skills provide good examples: we know how to ride a bike but we would hardly know how to put that skill into words. In teaching it to our children, we avoid books and manuals choosing instead personal teaching, trying to show them how it is done.

Although the importance of tacit knowledge is widely recognised in many human activities, its transmission is only possible through direct contact between the learner and the teacher, because it is knowledge that cannot be encapsulated, for example, in an algorithmic form. The focus on the *method* that accompanies the traditional and universalist practice of technoscience does not give due importance to tacit knowledge, although the latter is crucial to the practices of technoscience.

Overflows (instead of framings)

Plans/models necessarily make a series of assumptions about the world on which they propose to intervene. Such assumptions are related to how they represent this world, that is, plans/models correspond to a certain selection that operates in this world. This selection produces a simplification, a reduction of complexity, without which they could not acquire "generality" or "universality". We call *framing* this operation through which what deserves attention is detached from this world, that is, through which it is decided what belongs to the frame. However, at the same time, it establishes what stays "outside" the frame and, therefore, what does not belong to the world on which plans/models intervene. We call *overflow* everything that lies "outside" due to a framing operation [12].

Indeed, in any framing, there is an overflow. In other words, if some form of facing complexity is necessary so that it does not collapse on us (and therefore we will always need some level of framing), yet complexity is not apprehensible at once (and therefore there are always overflowings). If framing is a measure of success of a plan/model, overflow indicates the resistance to framing. However, it is through the overflowings that we can better know which world the plan/model refers to, and, therefore, verify the pertinence of the applied plan/model.

In terms of SE, models make several assumptions: organizational, cultural, social, environmental, architectural, labor relations, to mention a few examples. Such assumptions are "naturalized" as the practice of forcing an entity (corpora-

tions, public institutions, small businesses, etc.) to adapt its resources (hardware, software, engineers, etc.) to the proposed model becomes more usual, then safeguarding the model from any "guilt", which should be exclusively attributed to its misappropriation. It can be said that the world is "blamed" and the model "absolved", so that the framing it accomplishes is kept intact. Another option, the one we propose, is to "absolve" the world so that we can learn more about the model applicability through its overflows. If a model's success speaks well of the excellence of its assumptions, its failure speaks louder about something far more interesting, the world in which we live.

In proposing the indissociability of the "technical" and the "social", the "content" and the "context", the sociotechnical approach opposes the *normative desire* - that of simplifying; of establishing the norm, the model; of planning; of universalizing; of producing similarities – to the *descriptive desire* – that of describing in detail; of particularizing; of locating; of specifying; of producing differences. It is worth remembering that, when choosing locality and specificities, a Brazilian desire comes into scene, one of recovering our circumstances, and thus establishing a truce with circumstances that are eminently from abroad.

HOW TO PROCEED? FIRST SPECULATIONS ABOUT POSSIBLE PATHS

Against the "best practices" of SE plans/models, we have just arrived at the conclusions without any "proposal of solution" in hand. However, HERBSLEB [2, p. 26] warns that

in software engineering, we have a strong if unfortunate tendency to think that every paper should show a practical result that is immediately useful. [...] I think it is quite reasonable to expect programs of research to lead to practical results, but we need to spend some time and energy understanding how things work.

Following Herbsleb's suggestion, the present article has exclusively the pretension, which is already an ambitious one, to problematize the current practices of SE, whereby "non-technical factors" are "expelled" from their disciplinary reach, or at most, treated as second-class "aspects", always subordinated to "technical" framings. Thus, we see no alternative but to strive for a closer relationship with the human and social sciences, materializing a necessarily interdisciplinary movement which, in our view, is the only way out to tackle the pressing issues of software development for which SE, if preserved in its disciplinary "exclusivity", has no way to deal with.

In this sense, our proposals refer not to this or that "concrete solution", but to the very reconfiguration of the research agenda in SE in order to include new theoretical and methodological instruments coming from other areas of knowledge.

From the perspective of anthropology and history, we propose that the research in SE privilege:

Thick descriptions

In the first chapter of his *The Interpretation of Cultures*, the anthropologist Clifford Geertz discusses the work of the ethnographer. Briefly, his position is that the ethnographer's goal should be to observe, record, and analyse a particular culture. More specifically, he/she should devote him/herself to the interpretation of signs in order to reach their meanings in the midst of the culture in question. Such an interpretation must be based on what he calls the *thick description* of a sign, by which it becomes possible to apprehend all his meanings. His example [13, pp. 6-7] on *"contracting the eyelids"* clarifies the point. When one person blinks his/her eyes, is he/she only *"rapidly contracting the eyelids"* as it would appear in a *thin description*, or would it be, for example, *"the conspiratorial signal to a friend"*, a characterization only possible through *thick description*? It is through the differences between both descriptions that lies the object of ethnography,

a stratified hierarchy of meaningful structures in terms of which twitches, winks, fake-winks, parodies, rehearsals of parodies are produced, perceived, and interpreted, and without which they would not [...] in fact exist, no matter what anyone did or didn't do with his eyelids [13, p.7].

Ethnography is thick description. [...] Doing ethnography is like trying to read (in the sense of 'construct a reading of') a manuscript – foreign, faded, full of ellipses, incoherencies, suspicious emendations, and tendentious commentaries, but written not in conventionalized graphs of sound but in transient examples of shaped behavior [13, pp.9-10].

Through *thick descriptions*, Geertz hopes that the more detailed comprehension of signs will establish or broaden the dialogue between diverse cultures. In the case of SE, we can understand, at a first instance, that the dialogue to be established or expanded is that among the cultures of those who will study the implementation of software development plans/models (the ES ethnographer), those who adopt these plans/models in their professional routine (the practitioners), and those who conceive and disseminate them as the "best practices" in SE (researchers, teachers and consultants). However, a second instance should be highlighted: the dialogue between the cultures of producers of plans/models (mostly North Americans) and those of their consumers, among them, us, the Brazilian software engineers. A thick description applied to SE not only has the ability to elucidate what SE practices actually consist of, but also what are the tensions and asymmetries arising from the adoption in Brazilian institutions and corporations of plans/models that were not originally designed to their particularities and spec-

ificities. Describing the practices of software development in our country is, in our view, an indispensable way to conceive the usefulness of Brazilian software engineering in developing knowledge in SE appropriated to (and problematized from) local needs.

The "denaturalization" of models and artifacts troughs their histories

The historical analysis of a SE development model/process, in addition to providing lessons for the development of new software production technologies [14], is itself an indispensable contextualization for understanding the assumptions and the scope of the promises of a particular model/process. A model/process without history becomes an "universal", suggesting that it can be applied in the same way, and with the same effects, anytime, anywhere. Therefore, a solution, if isolated from the historical circumstances of its conception – what problems were originally tackled, what effects were then intended, who were the beneficiaries, etc. – becomes a "natural solution". Our proposal is to go hand in hand with the history of the proposed solutions to/from SE, that is, to "denaturalize" them, seeking, through their historicity, to establish parameters that allow to evaluate their circumstances of origin in view of the actual circumstances of their use. Indeed MAHONEY [15, p. 9] puts historians and software engineers side by side in their interest in the history of software engineering:

> it may help to think of historians and practitioners as engaged in a common pursuit. Both seek a history for software engineering, though not for the same purpose nor from the same standpoint.

To the most patient and curious reader who has honoured us with his/her attentive reading up to these last lines, perhaps there remains one last question: after all, am I reading an article on SE? To answer this question, we offer as our last words the providential considerations of James Herbsleb [2, pp. 26-27]:

> Interdisciplinary research may seem completely beyond the pale. In some environments, I have heard people asking the question, of one research program or other, 'But is that really computer science?' Many people spend significant time and energy worrying about this question, apparently. The best answer I've heard was given by my colleague Randy Pausch [...] 'Do something great; we'll decide what to call it later!'

REFERENCES

[1] CUKIERMAN, H., TEIXEIRA, C.A.N., PRIKLADNICKI, R., 2007, "Um Olhar Sociotécnico sobre a Engenharia de Software". In: RITA - **Revista de Informática Teórica e Aplicada**, vol 14, no. 2, pp. 199-219.

[2] HERBSLEB, J. D., 2005, "Beyond Computer Science". In: ***Proceedings of the 27th International Conference on Software Engineering (ICSE)***, St. Louis, Missouri, EUA, pp. 23-27.

[3] BOEHM, B., 2006, "A View of 20th and 21st Century Software Engineering". In: ***Proceedings of the 28th International Conference on Software Engineering (ICSE)***, Shangai, China, pp.12-29.

[4] LATOUR, B., 2000, ***Ciência em Ação: como seguir cientistas e engenheiros sociedade afora***. São Paulo, Editora UNESP.

[5] SUCHMAN, L., 1987, ***Plans and Situated Actions: the problem of human machine communication.*** Cambridge University Press.

[6] TEIXEIRA, C. A. N., 2006, "Algumas observações sobre os vínculos entre a Engenharia de Software e o pensamento moderno.", In*:* ***Workshop Um Olhar Sociotécnico sobre a Engenharia de Software (WOSES)***, 2., Vila Velha. Anais... pp. 39-50.

[7] HANSETH, O.; MONTEIRO, E., 1998, ***Understanding Information Infrastructure***. Manuscript. Available at <http://heim.ifi.uio.no/~oleha/Publications/bok.pdf>. Accessed on: 28 abr. 2018.

[8] DAHLBOM, B., MATHIASSEN, L., 1993, ***Computers in Context: The Philosophy and Practice of Systems Design.*** Oxford, NCC Blackwell.

[9] PHONGPAIBUL, M., BOEHM, B., 2005, "Improving Quality Through Software Process Improvement in Thailand: Initial Analysis". In**: Proceedings of 27th ICSE**, Workshop on Software Quality.

[10] ARNOLD, M., 2003, "Systems Design Meets Habermans, Foucault and Latour". In: CLARKE, S., et. al (eds.), ***Socio-Technical and Human Cognition Elements of Information Systems.*** London, Information Science Publishing, pp.226-248.

[11] METCALFE, M., 2003, "Concern Solving for IS Development" In: CLARKE, S., et. Al (eds.), ***Socio-Technical and Human Cognition Elements of Information Systems.*** London, Information Science Publishing, pp.135-151.

[12] CALLON, M., 1998, ***The Laws of the Markets***. London, Blackwell.

[13] GEERTZ, C., 2000, ***Interpretation of Cultures***. Basic Books, New York.

[14] PORTO DE ALBUQUERQUE, J., 2006, "Por uma Perspectiva Sociotécnica no Desenvolvimento de Sistemas de Computação: o exemplo do Modelo Mikropolis", In: