Applying Problem Based Learning educational method for improving Human-tech competencies in Computer Engineering students: a research proposal

PhD candidate: Nuno Valero Ribeiro[1]

Escola Superior de Tecnologia de Setúbal
Instituto Politécnico de Setúbal
2910-761 Setúbal, Portugal
nuno.ribeiro@estsetubal.ips.pt

Abstract. This paper resumes the background theory of the likewise entitled research project. The project aims to give a contribution to software programming quality improving "Human-tech" competencies in Computer Engineering students as a means to prevent, or at least avoid in a great extend, the rate of unsuccessful software implementation projects. We are specially interested in researching what Human Factors competencies must be profiled in Computing Curricula outcomes that may contribute to better prepare students as "Human-tech" experts. We will apply Problem Based Learning educational method for delivering those competencies to students. We believe it is possible to do better than what as been done, to have a better degree of adequacy between the Human user and the software he uses for his/her activity. All background theory that support the axiomatic principles of this research project is explained in the first section. Then the project is outlined as well as its plan and expected outcomes and contribution in the following sections.

1 Introduction and Background

1.1 Information Systems

Summary:

1. IS's: general considerations and examples
2. Could *prevention* be a better strategy?

Current Status Implementation of Information Systems (IS's) (computer supported) in organizations do not always correspond the set of initial expectations. My experience as computer practitioner and as a computer engineer as led me to frequently ear complains about IS's. Each of these complains was analyzed, in a quest for the origin of these bad matches, searching for possible causes that would explain this lack of results. At first sight one gets the traditional explanations: over estimated expectations, bad design plans, organizational resistance

to changes, poor implementation, deficient planning of resources, lack of qualification, etc. But, digging more deeply, the following statements turn out to be the common link valid in all these cases:

> 1. Information Systems (computer supported) are typically considered as the "Holy Grail" that will solve all problems of an organization.

Instead, they become the problem itself. Time and money are consumed, attempts to fix the problems are expensively made, workers despair trying to use it, client complains tend to be excused blaming the computer, managers ask for solutions and developers try their best.

> 2. End users are neglected (specially) in the planning and development phases of Information Systems computer supported.

The lack of consideration and participation of end users during the planning and development phases of the project typically leads to complains in the implementation phase. If IS end users are not consulted, their particular needs (vs. organizational) are not predicted or considered, and their jobs will not improve with the implementation of the IS. Structural coupling between IS's and its end users should always be considered as a *must have* requirement. Information Systems are typically designed with the goal of improving organizational efficiency in mind and neglect what can be the end user needs to operate properly with its support. Naturally, deceptions may occur.

Examples Different aspects of the problems described above are illustrated in the following examples taken from situations I've come across with:

1. The "CBC" case: CBC is an enterprize whose business is assuring quality demands in several fields. CBC has experts whose main job is to visit different companies, geographically dispersed, and write a report as the result of their visit. CBC experts began complaining when started to use the IS for the task previously hand written. They complain having to spend more time now, and effort, dealing with the CBC's IS than practicing their expertise competencies. This was not what enterprize managers wanted when implemented the IS, nor what experts aspire to. Statements like: "the IS is inadequate to our job practices" or "the IS is a terrible headache" are generally heard coming from the expert. Similarly, this situation occurs in other organizational environments, like, for example, in health care where nurses or doctors spend more time operating the computer than with attending the patient, as they should, since they were trained for it.
2. The "STB" case: STB has an IS that supports the normal business processes like, for instance, requesting a room reservation. Its quite simple: the user just has to fulfill an IS data form about the event, hour, day, etc., clicks on a button and waits for the answer. The answer appears as a warning in

the IS front end after a while. Nothing new. The problem occurred when the IS user was at the room door, with the confirmed reservation ticket at hand, which was produced via the IS, and soon realized that the room was occupied by someone else for another event. He was totaly embarrassed, and deeply ashamed. He had asked VIP's to come, speakers, who had previously confirmed their agendas and carefully prepared the conference presentations, plus friends and colleagues he had invited to attend the conference. Worst of all, there was no other room available. He asked for responsibilities since he did every action correctly and the reservation was validated. It just happened that the room was occupied by an event that was not registered in the IS. Why? Because the occupying event was considered so "important" (and needed so many rooms) that fulfilling IS forms was just a waste of time. The work load of the administrative functionaries in the weeks before was so big that instructions were given for not to worry about "infocratic" work and it was assumed that everyone knew about the big event. Well, unfortunately, not everyone knew.

3. The "SIT" case: SIT is an IS that supports the ordinary actions of several participants in a distritbuted organizational process. The documents involved in the process, and stored in the IS, are classified and have high security requisites like personal electronic identification cards for user authentication. Thus, the IS has an uncommon authentication mechanism. One of the key participants in a business process had a health problem, leading to a health license and a big period of time absente in the process. A replacement had to be done in the process. The problem was to deal with the security constrains of the IS. How to replace the user that fulfils a function within a process since it has already began? The IS administrator started by assigning the new user to the process, but soon realized that a new user could not access previously stored documents owned and registered by the replaced user. He tried to override the previously assigned user for these documents and assigned them to the new one. But, someone reminded him, he would put in stake the true authorship of the those documents. Actually, he did try, but the IS didn't allow it. Well, after spending hours dealing with the problem, phoning experts, etc., a decision was made. It was allowed for the new user to access the IS with the authentication privileges of the former user, handling him the former authentication "personal" card. This required a personal trust relationship among both of them: the former user had to lend the authentication card to the new one, and, the new one didn't mind "signing" his documents as another person. An odd situation considering the security requisites of the present IS.

1.2 Preventing instead of Correcting

Technicians strive for designing and implementing IS's capable of supporting all organizational business processes. The major drive force is a political and managerial desire to control and operate everything within the organization. This is mainly due to the globalization and economical trends that lead organizations

to become off-shored and to have *on the fly* data for managerial purposes (see [12] for complete explanation of the subject). Adequation of IS's to this need is thus imposed from the top management as a necessity and does not come from the bottom work force of the organization typically. Users are generally forced to adapt their usual procedures to become compatible with the planed procedures implemented via the IS.

On the other hand, life is full of unexpected and uncontrolled events and organizations are living cells. As technicians improve and add functionalities to the IS, in order to support new business processes or to solve unexpected problems, the IS starts to endlessly grow and soon becomes an impossible-to-use-and-manage Babel.

This is due to the typically adopted approach to deal with the problem. The solution is more focused on a *detection and correction* strategy. Thus, a never ending story of maintenance and post-development efforts are generated. Focusing the solution on a *prevention* strategy, or, at least, *avoiding* mismatches at a greater extent, although initially more expensive, could be rewarding at the end. We believe that situations like those exemplified above could be prevented in the designing phase of technology if end users were more taken into account.

In order to explore solutions based on *prevention* or *avoiding* strategies one must first break the *sine qua non* conditions of the unsuccessful implementations of Information Systems. Development approaches in this engineering area still ignore its human and social connections, and implications, as pointed out in [11].

1.3 Software Engineering particularities

Summary:

1. Software Engineering vs. other engineering disciplines
   (a) The immaturity of Software Engineering field;
   (b) Base Sciences that support Software Engineering;
2. Axiomatic set of principles for Software Engineering

When comparing Software Engineering with other traditional engineering fields (like Mechanical Engineering, Electrical Engineering, etc.) we must agree that: 1. Software Engineering is more recent and, therefore, still in an "adolescent phase", full of experimental practices; and 2. Software Engineering is intrinsically different in the sense that traditional engineering base sciences do not apply to the field (like cinematic physics or electromagnetic physics).

1. The immaturity of Software Engineering field.
   In traditional engineering fields those who plan and design the technological artifact (for instance, a house, a bridge, an electronic motherboard or electrical plant) become specialized in that area, i.e. *planning and design*. Their concern is different from those who are concerned with building the artifact. Moreover, those who check if the technological artifact is being built accordingly to the initial planned design, or those who test it, may represent another group of specialized engineers.

Software Engineering does not yet benefit from these separation of concerns and specialization, at least, at the extent that traditional engineering areas benefit. Software Engineering specialized offices for validating initial design plans, or for testing software developed by third parties, are to much expensive and the procedure is not applied. At least, the procedure is not practiced as it is in other traditional engineering fields. Most of times, in software business, not even the owner knows what he really wants with the implementation of an IS or software technology in his organization.
Traditional engineering fields have become specialized through time. Design plans and methodologies for building their technological artifacts (like electronic maps, architectural plants, etc.) became consensually accepted and normalized. Generally those include accurate calculus and drawings, or procedures. This is due not only because of its maturity but also because those engineering areas are an application of exact sciences like physics.

2. Base Sciences that support Software Engineering;
Base sciences are those from which the *application of* leads to a particular engineering field. The materials "crafted" by software engineers are not tangible. As the name implies they're soft! They're more related with information, people and organization, then with physical phenomenons studied in Electromagnetism, Electronics, Mechanics, etc. This fact makes Software Engineering a different "kind" of engineering field. We must take into account theories coming from human sciences like social, business, organizational sciences.

As we said, Software Engineering deals with people, organization and information: ways of storing, processing, changing, organizing, retrieving, reaching, displaying, arranging information, and so on. And, Software Engineering doesn't have, at the moment, a commonly accepted conceptual way for planning or designing IS's or software. Thus, we may conclude that:

Software Engineering has, in its foundation, exact sciences as well as human sciences.

Nevertheless, human sciences theories have been experimented in methodological approaches for designing IS's. For example, DEMO [3] methodology — Design & Engineering Methodology for Organizations — adopts a basic pattern of coordination as the building block for designing all business processes. This pattern is based on Habermas's *Theory of Communicative Action* [8] borrowed from linguistics. Or the MEASUR Organizational Semiotics based methods covered in chapter 4 of [13].

This methodologies are focused on a basic principle of mapping the organization's business processes in an *as-is* fashion for designing the IS at as a starting point. They are necessary as a starting point. Afterwards, the organization evolves for itself, demanding changes to the IS.

Other patterns of conversation do exist in organizations, and therefore, not all aspects of interaction among persons in an organization are supported via these

approaches (we do not say they should be, just stating the fact). For example, a person may spontaneously interact with another person emitting a judgement about some piece of work. This pattern is not captured, unless if making part of a well-known business process. Although it is not part of the organization's regular business procedures, it might influence the organization evolution. It may even begin an innovative process in the organization and thus a future correction and development in the IS that supports it.

1.4 The User and the Organization

An organization is fundamentally a social environment as it is recognized by Dietz's in [4]. Since organizations are "living cells", narrowing the design of IS's to a particular or momentary view of the organization could represent a limitation to its own evolution. Structural coupling between the IS (that supports the work being developed in the context of an organizational activity), and its users, should be as extensively achieved as possible. In other words, as a driver may adapt the car seat to his personal physical stature, the user of an Information System should be given the possibility of adapting it to his functional roles and usage within the organizational context.

This does not typically happen nowadays: IS's have the same generic front-end and functionalities for all users, regardless of their particular functional needs in the organization. The access to the IS functionalities is not designed taking into account the user as an individual person, with particular needs and roles within the organization. The design is centered on a generic mechanical point of view of the organization, where the user is taken as a *piece* of the organization's complex mechanisms. The vision contained in the (prevention) strategy for a solution is taken from Vicente in [15]:

> Its necessary to tailor the design of technology to people, rather than, pushing people to adapt and decipher technology.

As Robert Brigs explains in [1] *a good theory is a* model *of* cause-and-effect *to explain some phenomenon of interest. Every technology presumes a cause-and-effect. Every technology is built to improve some outcome.* Therefore, the question is: what outcome do we which to improve? The outcome we which to improve with Information Systems or software introduction in organizations will set the phenomenon of interest for our research. The expected outcome to improve with technology introduction in organizations must thus be clear.

1.5 What are computers for?

Since Information Systems are deployed with the support of computers nowadays, this leads to the question: what is the purpose of implementing an IS

supported by computers in a organization? That, in turn, leads to the fundamental question: *What are computer for?*[1]

The answer to this question is not an obvious one although it may seem easy to give an answer since computers are widely used by many people and in many contexts nowadays. We have run the experience of asking the question to different classes of Software Engineering students and the fact is that, among them, we have collected a set of totally different answers to the question [see collection of data *1]. Thus we may say that there is no (common) understanding about the purpose of computers.

The user of technology should have a clear idea of the purpose that it has. This applies not only to the final user, in strict sense, but, and even more seriously, to software engineers, and computer programmers, and owners, since their responsibility is bigger when building, developing and owning IS's.

In the introductory chapter of the book *Organized Activity and its Support by Computer* [9], Holt discusses and formulates that

> Computers are for reducing the effort of carrying out organized activity.

assuming a broad sense of the terms "computers" — any computational device either connected or not in networks — and "organized activity" — a human universal that exists wherever and whenever people exist.

Furthermore, the author also points out:

> The practical side of the question "What are computers for?" is the question "How should computers be programmed?".

Thus, the more clearly we understand "*What are computers for?*" the more efficiently we obtain the expected outcome of using computers. Particularly, if we are builders of Information Systems, or programmers, we should have the necessary tools for facilitating this perception.

Still, many programming projects are developed by teams of programmers who have to deal with, and understand, complex human and organizational problems without the necessary tools, and owners who will never understand the programs that are written and operated on their behalf.

Once computers are meant for helping people carrying out an Organized Activity (OA for short), the effort we spend performing an OA, when supported by computers, should decrease. Otherwise computers wouldn't be fulfilling their purpose. Since operating computers also requires human effort, there should be a tradeoff, and an ideal point, between the effort spent when using computers to support an OA vs. the benefits or outcomes achieved with the use of computers. And thus, this may bring a conflict of interests between the organizational goal when implementing an IS and what the end-user needs.

[1] This issue was brought to me by Anatol Holt during is stay as invited professor in Technology School of Setúbal, Portugal, with whom I had the opportunity to have long and challenging discussions about computers, most of them based on his book *Organized Activity and its Support by Computer* [9].

1.6 The Mechanistic view of the world

1. IS's inflexibility to changes

Typically, software engineers have a mechanical view of situations. They are trained in programming. This implies pre-defining possible routes of programm execution. Their "natural" tendency is to build programmes that map the analyzed business processes, and its flow of execution, into the architecture of the IS in an "As-Is" fashion (read [2] for example). Exceptions to the foreseen flow of execution and generally prohibited. They do not think about how to cope with future changes or exceptions to the plan.

Attempts made in this direction become hard to maintain. For example, the approach in which TOA (Theory of Organized Activity) [10] is based, to map the organization, assumes that a user of an IS is a tuple [person+function]. Either persons or functions are constantly changing in organizations. Plus, the external environment also imposes changes to organizations (legislation, clients, natural phenomenons, etc.). Changes become difficult to implement in an IS designed in this fashion. Unexpected events are difficult to handle. We have adopted TOA in the development of an IS [14]. Soon it became difficult to manage. It constantly demanded updates to the functions that were fulfilled by the persons of the organization. We realized that the only possible solution was to ask the employees themselves to adapt their profile in the system. This implies a more "As-the-user-needs" fashion, or approach, to design than the "As-is" fashion. In the initial plan, a responsible settled the employees profile according to the function(s) they were fulfilling in the organization at that moment. We were thinking only about the user as someone performing a function and neglecting the user as an individual person capable of deciding for himself. We were adopting only a mechanical point of view of the organization and neglected the humanist point of view.

Detecting and correcting the IS to cope with changes requires big post-development efforts. The construction of an IS capable of coping with the pace of constant changes in the organization is a challenge.

> Since the major vehicle of change in organizations is the human being, centering design decisions in the human person is mandatory to achieve flexibility.

Software engineers must take into account that organizations are "living cells" that evolve for themselves. Thus, IS's should be flexible enough to allow constant implementation of changes in their construct. This approach contrasts with the "box" fashion of selling software or "package" approaches. IS's should be though and conceived considering a more (not completely planned) "Lego" approach. The user should choose his front-end interface from a menu of implemented functionalities.

Letting the user shape his front-end and set-up the IS's functionalities that he needs is likely to be the most suitable way of coping with the pace of changes

that occur constantly in organizations. IS support should be tailored and shaped for that particular user but this may only be achieved if we give the possibility of choice to the user. If this becomes a reality, the "structural coupling", or degree of fitness, between the outcome achieved through the support of the activity by means of an IS and the user needs will increase. A person, although taken as a user of an IS that is fulfilling a function in the organization, has interests, needs and will of his one.

> To achieve a greater degree of fitness, when designing an IS, we must take into account the organizational goals as well as the user personal interests.

1.7 Tailoring Technology for People

In the book *The Human Factor: revolutionizing the way we live with technology* [15], Kim Vicente points out that scientific knowledge has been divided into two big groups: human sciences and technical sciences. Human scientists, when they look at the world, they focus primarily on people. Technical scientists have adopted a mechanistic view: they look primarily on the hardware or software. He defends that design of technology must take into account the characteristics or needs of people who will be using it because, people and technology interact and exist, side-by-side, in the real world. We must resist adopting a partial vision of the real world for orientation in the process of designing technology.

Human-tech Oriented Design Vicente suggest the compound word "Human-tech" to remember us that people and technology are both important aspects of the system. "Human" comes first to remind us that we should start by identifying our human and societal needs, not by glorifying some fancy widget in isolation; "tech" is a means, not an end in itself, so it comes second. He proposes five human aspects that may guide the technology designer in his work. These are: physical; psychological; team; social; organizational; political. In each one of them, human sciences have a lot of possible contribution for helping design decisions. It is not a rigid separation of concerns, may suffer adaptations. If we consider the human characteristics that are relevant to the specific design problem we want to solve, everything becomes much, much simpler.

Vicente states that adopting a context-specific, problem-driven approach narrows down the amount and kind of knowledge that we need to consider to find an effective design solution. The adopted solution must respect the physical, psychological, etc., characteristics of the people that will be using it in order to build an harmonious relationship between people and technology.

> Human-tech oriented design may serve as a guide for the desired "structural coupling" between technology and its users.

1.8 Problem Based Learning

In order to train programmers for the design of software solutions based in context-specific, problem-driven approaches, one must chose the most probably correct methodology and test if it is effective. Applying PBL (Problem Based Learning) seems to be an adequate educational method for delivering "Human-tech" competencies to programmers. Accordingly to the UCPBL Centre for Problem Based Learning located in Aalborg [6], this method is adequate for changing challenges in educational engineering schools who want to develop a more student centered solutions focused in the learning process and on the development of new competencies such as "Human-tech" in computer engineering students.

The pedagogical principles and historical background of PBL may be found in a special edition about PBL and ICT (Information and Communication Technology) [5]. Like the approach brought by the introduction of this educational method in the McMaster medical school of Canada [7] we believe that, similarly, "Human-tech" should be learned in practice focusing on the Human user of computers and on his/her requirements. By systematically analyzing the user problems, students will formulate questions, search for information lacking to problems, select their learning goals. The objective is to train the student to act and think as a Human-tech expert. The experience obtained by the student is expected to become meaningfully in his future as a professional.

The model integrates a number of pedagogical principles: problem-orientation, inter-disciplinarian, participant control, exemplary project, teamwork, and action learning. However, the model is practiced in various ways adapting it to local conditions, subject matters, skills of students and supervisors, etc. Thus, adaptation of the model to local conditions and "Human-tech" competencies to be learned is required.

2 Research Project Description

2.1 The Problem

To many software applications nowadays have a bad "structural coupling" with the persons who are using them. According to Vicente (previously cited), this is essentially due to the lack of "Human-tech" competencies in those who design and develop technology. As Vicente points out: "when the wizards of technology design their gadgets many consideration of human aspects are neglected, or even, not considered at all."

Programming is not an easy task and programmers are heavily trained in technical skills. Generally, computer engineering degrees do not have special courses for qualifying their students in "human" competencies. Those competencies are typically said to be transversally taught and are typically neglected comparing with technical specialized competencies. There is a necessity to improve "Human-tech" competences/skils deliverance to Computer Engineering students as a means to improve software production quality.

2.2 Research Question

What "Human-tech" competencies and skills are needed for Computer Engineering undergraduate students in order to improve "structural coupling" between persons and computers. What results are obtained when applying PBL educational method for improving those "Human-tech" competencies in Computer Engineering undergraduate students?

Sub-questions What is the "Human-tech" profile of competencies for a computer engineering undergraduate student?

How to evaluate if a student have learned those competencies?

Does PBL proves to be an adequate method for delivering "Human-tech" Competencies in Computer Engineering degrees?

What are the specific characteristics od the adopted model of PBL in the tested scenarios?

2.3 Problem Significance

If we have better "Human-tech" programmers we should have better adequation of software design and development to organizations and users. More efficient employment of software programmers resources. More efficient return of investments in software development projects.

3 Research Plan

3.1 Research Work Plan

Start: OUT 2012 ... End: OUT 2015

tasks and outcomes

1. Literature study and field study
   (a) Actual profile of Learning Outcomes in Computer Engineering Curriculum
   (b) Desired situation for Human-tech Oriented Design of software
   (c) Formulate hypothesis comparing both
2. Field research for construction of learning outcomes evaluation
3. Proposal of a PBL based model for enhancing Human-tech learning outcomes
4. Case studies
   (a) Apply and test the model in real situations
   (b) Analysis of results
   (c) Processing of the outcomes
5. Reporting

3.2 Research Outcome

The expected outcomes of the project will consist on the following:

1. a list of Human-tech Computer Engineering learning outcomes;
2. a PBL based model for learning Human-tech competencies;
3. a method for evaluating Human-tech learning outcomes;
4. an assessment of the model provided through its application on Computer Engineering curricula;

Several reports will be produced, as indicated in the work plan, some of which are expected to be published in international conferences and workshops.

3.3 Validation Process

The validation of the outcomes of this project are not possible in laboratory due to the essential nature of a learning environment that cannot be artificially reproduced. Thus, a possible validation may be through applying the model in a set of case studies, i.e. Computer Engineering curricula, and making an assessment, confronting the results obtained with reality reported.

3.4 Organizational Context

Name	Role	Organization
Prof. Carlos Pampulim Caldeira	Supervisor	Universidade de Évora
Prof. Joaquim Filipe	Co-Supervisor	Instituto Politécnico de Setúbal

3.5 Research Value

Are those outcomes worth the effort? Whose work might be more effective? Whose life might improve?

To many times although, the "invisible" costs of a unsuccessful software piece, or IS implementation, are neglected, specially for the human user side of the coin. We which to give a contribution to the improvement of the rate of successful software implementation, focusing specially in its adequation to the (Human) user and organization. We think its be possible to prevent, or, at least minimize or avoid, the rate software implementation mistakes that, sometimes, jeopardize the success of technology innovation in organizations. We believe it is possible to do better than what as been done, to have a better degree of adequacy, to avoid unsuccessful implementation of software.

References

1. Briggs, R.O.: On theory-driven design and deployment of collaboration systems. International Journal of Human-Computer Studies 64(7), pp. 573–582 (July 2006)

2. Castela, N., Tribolet, J.: AS-IS Continuous Representation in Organizational Engineering. In: ICEIS 2008: Proceedings of the Tenth International Conference On Enterprise Information Systems. vol. I, pp. 371–374. INSTICC (2008)
3. Dietz, J.L.: Enterprise Ontology: Theory and Methodology. Springer-Verlag Berlin (2006)
4. Dietz, J.L.: The Deep Structure of Business Processes. Communications of the ACM 49(5), pp. 59–64 (October 2006)
5. Dirckinck-Holmfeld, L.: Innovation of Problem Based Learning through ICT: Linking Local and Global Experiences. International Journal of Education and Development using Information and Communication Technology 5(1), pp. 3–12 (2006)
6. Enemark, S., Kolmos, A., Moesby, E.: Global network on engineering education research and expertise in PBL. Samlignsnummer för enstaka enskilt utgivna arbeteb (2006)
7. de Graaff, E., Kolmos, A., et al.: Problem Based Learning. final report of the Special Interest Group (SIG) B5, TREE (Teaching and Research in Engineering in Europe) Thematic Network of SOCRATES/Erasmus programme of the European Commission (August 2007), uRL, http://www.unifi.it/tree/dl/oc/b5.pdf, accessed October 2011
8. Habermas, J.: The Theory of Communiative Action: Reason and Rationalization of Society. Polity Press, Cambridge (1984)
9. Holt, A.W.: Organized Activity and Its Support by Computer. Kluwer Academic Publishers (1997)
10. Holt, A.W.: The "Organized Activity" Foundation for Business Processes and Their Management. In: Business Process Management, Models, Techniques, and Empirical Studies. pp. 66–82. Springer-Verlag, London, UK (2000), http://portal.acm.org/citation.cfm?id=647778.757179
11. José Cordeiro and Joaquim Filipe and Kecheng Liu: NOMIS - A Human Centred Modelling Approach of Information Systems. In: Proceedings of the 4th International Workshop on Enterprise Systems and Technology. Athens, Greece (2010)
12. Levy, F., Murnane, R.J.: The New Division of Labor: How Computers are Creating the Next Job Market. Princeton University Press (2004)
13. Liu, K.: Semiotics In Information Systems Engineering. Cambridge University Press (2000)
14. Ribeiro, N.V., Delgado, J., Rolo, J., Lourenço, R.: Application of Theory of Organized Activity to a Real Case Study. In: Cases and Projects in Business Informatics. pp. 167–181. Logos Verlag Berlin, Berlin, DE (2006)
15. Vicente, K.: The Human Factor: revolutionizing the way we live with technology. Vintage Canada (2003)