QUALITY IT - HAS THE TIME COME?

Bogdan Pilawski

Bank Zachodni WBK, Strzegomska 8-10, 53-611 Wrocław, Poland
e-mail: bogdan.pilawski@bzwbk.pl

Selected thoughts on "Social Thinking – Software Practice" collection of papers [Dittrich_2002]

Abstract

At present there are numerous tools and methodologies available to ensure the quality of software products, from their design stage up to the way they are run while in production. Despite all this the IT as a whole is far from being perceived as quality service.

There seem to be two reasons at least for that: one is the lack of importance attached to the area of requirements refinement, the other is the low quality of data, which the IT systems allow to creep into the resources they store and process.

This paper concentrates on the ways, the requirements regarding IT system are discovered in detail, and converted into IT solutions. Numerous papers on that subject have been presented in 1999, during the seminar in Schloss Dagstuhl, in Germany, and then, after some adjustments, collected in the [Dittrich_2002] book.

This paper attempts to review this collection of papers, and to discuss further some of the questions raised there.

1. Introduction

When this paper will become published, it will be almost exactly one year since *Harvard Business Review* published Nicolas Carr's article "IT Doesn't Matter". Its title is obviously and deliberately exaggerated by purpose. Its contents attempted to assign IT with a role of similar other areas of practice, since long present in the life of many communities. Carr's paper also attempted to deprive the IT of the role of an area of special importance and deserving special attention, which it enjoyed since its origins.

According to Carr we're right now witnessing the IT reaching the level of importance for years enjoyed by electricity and telecommunications. No one would attempt to question the positive role of those two areas, but at the same time they are neither subject of special interest nor do they expect special treatment, simply by their very existence. Paradoxically, those areas are becoming subject of special interest only when there are some disruptions in the services they provide or they're becoming unavailable at all.

The title of Carr's paper is somehow provocative, since it requires the reader to perceive the IT as a one whole, and with wider perspective. At the same time one needs to acknowledge, that the level of service and also that of availability and of reliability of IT services is still far away from the level for long provided by telecom operators and suppliers of electricity.

2. The IT quality issues

The lack of easiness of use of IT services leads directly to conclusions of similar kind. Even the services of Internet simplicity are in fact much more complex and do require its users to have a certain level of specialised knowledge. The lack of such knowledge in many cases makes a more sophisticated use of those resources difficult or even impossible (what finally confirms Carr's thesis).

Since the time the IT grew out of its pioneering years, many of its sub-areas became subject of intense theoretical research in many disciplines. That resulted in rise of multiple methodologies devoted purely to the creation and use of IT systems. After some time those methodologies have imposed their rules also on the act of IT systems and creation, gradually replacing individualised and individual solutions.

Initially this has been received as an attempt to limit creative freedoms. This concept however was accepted later, but not without reservations. The resulting changes were similar to those induced by industrial methods replacing craftsman manufacturing in the early 20th century. To stay within this analogy - the industrial production has also changed during recent years to highly automated production process, allowing for of tight quality controls performed after each single manufacturing operation to be dropped and replaced with techniques and technologies, which by their very nature superimpose the required quality level, independently of the quantities manufactured [Guess_2001].

The creation methodologies of IT systems are gradually getting closer to the level already reached by manufacturing industry. The path to that lead through the techniques of structured design and programming, then trough objective methodologies, to reach the stage of systems known as CASE[1] and Rapid Development.

Other results of that process are the formalised methodologies designed to guarantee a certain software quality level. The most widely known of those is the Capability Maturity Model (CMM), which originated from Carnegie Mellon University [Paulk_1993].

The continuous progress in the area of IT theory gradually relieved software creators of the task of programme code optimisation, and of the necessity to design their own internal workings of database systems. Also the ready made software is nowadays tested for errors and omissions with highly specialised software tools, making that process fully automatic, and allowing for multiple repetitions of exact situations, which lead to errors to be discovered in the first place.

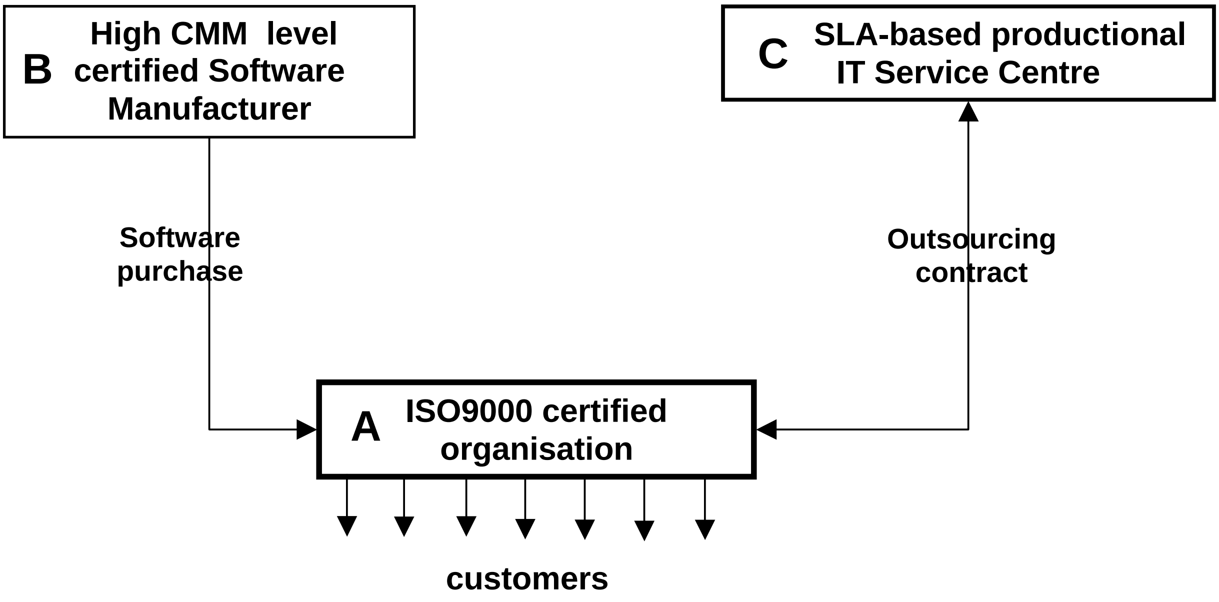
[1] CASE = Computer Aided Software Engineering, a software designed to facilitate the process of gathering user requirements, and to stage wise refine it by prototyping

The production use of computer software is usually subject to detailed agreements. These agreements regulate the service accessibility level required, its efficiency and the level of security of operation, to name just these few.

More to that - many organisations are now being certified for fulfilling the requirements of the ISO9000 family of quality standards, meant to guarantee the uninterrupted level of quality of services throughout such an organisation.

It is then possible to picture the situation (see picture 1), when an ISO9000 organisation (A) has purchased an IT system from another organisation (B), which created this system maintaining a certain CMM level. That system, in turn, is then ran by a computer centre (C), to which it was outsourced based on an high-demand SLA agreement.

The way the customers of organisation A perceives that situation, could be called as "**quality of IT**" view. This view not necessary reflects the high level of service provided by each of the organisations (A, B and C), and could be even critical or negative at all. That leads to the conclusion, that also there are some other factors affecting the value of an IT system, when seen as a one whole.


B
High CMM level certified Software Manufacturer
C
SLA-based productional IT Service Centre
Software purchase
Outsourcing contract
A
ISO9000 certified organisation
customers

Picture 1. Dependencies between IT system manufacturer and its purchaser, whit outsourced IT operations

One may name at least two such factors, and both of them stay well outside the quality assurance process as presented in picture 1. First of those factors is meant to ensure the IT system in question fits the detailed requirements of its future users. The requirements understood as both -

refinement of requirements[2], and also bringing the man-machine interface attributes as close to users' expectations as possible.

The task of defining the way, the IT system communicates with its users, is only a part of the process of requirements refinement, and should precede the process of system creation rather than result out of it. The multiple dilemmas related to that are since long observed in many areas, and they get even deeper while the systems, objects and devices used by people everyday are becoming more and more complicated. The nature of that problem is how to provide such an object with usability features expected by those meant to use it, and how to pass on the message, that the features in question are in fact present and at all available. More to that – if one object has many of those features available, the problem arises how to order them in a kind of sequence, to make any of them easily available, while ensuring they are reachable using intuitive thinking only.

The designers of objects in their practice usually do not undertake the effort, required to understand the requirements of future users of those objects in full. At the same time, those users are unable to express and communicate their expectations beforehand.

If to understand the category of computers (or even the IT systems) as belonging to objects, it turns up that those dilemmas and divisions are becoming even more visible. They are the substance of everyday effort of millions of computer users attempting to comprehend the secrets of how to use them in the first place, and to find some commonality between those rules of computer usage and their own comprehension of the outside world.

This problem does still exist: no satisfactory methodology has been set up to translate the real life rules and dependencies into the operational pattern of IT systems. The patterns, which could be felt as natural, and open to intuitive selection by users. The above does not mean there is no progress in that area – definitely some progress appears to exist, but put against needs, it is to slow, and to limited in scope and in effects.

On the software market, there are multiple tools available to aid the description and definition of the user requirements, and to automate the processes related.[3] Those tools however are unable to penetrate deeper into the sphere, where IT does converge with the social processes and phenomena, the nature of which the IT systems are expected to reflect. The solutions offered here by IT are continuously far away from the needs of reality, and there are no significant indications of foreseeable changes in the future.

3. To reflect what the user wants

So many papers and books have been written on weaknesses in solutions originally meant to facilitate the co-operation between humans and computers, and all that so far has been almost to

[2] the subject of mastering the requirements is for years an area of interest and specialisation to Suzanne and James Robertson; they authored many books and papers on that subject; see e.g. [Robertsons_1999]

[3] kept up-to-date and comprehensive list of such tools is available on the web pages of Atlantic Systems Guild; that list contains also a short characteristics of every tool presented, and basic information about manufacturers

no effect. The progress in that area seems to be slow and limited. Multiple, sometimes radical, and contradictory among themselves, views of many authors on that subject are presented in the book [Diettrich_2002]. That book does not bring a ready recipe of how to refine the requirements to IT systems, but it does create the picture of how complex the problems arising at the touching point of social thinking and the practice of IT system creation could be.

The multitude of opinions presented there does not make the subject any easier on one hand, but on the other, it confirms there is no uniformity of opinions among specialists interested in having the problem resolved. And that, in turn, puts a solution of some kind even further away. The above does not mean the authors are superficial in handling the substance of their research. It is even to contrary – the area of their investigation gets so wide in places, that the reader is faced with disciplines like ethnography, and economics, and feminism, and postmodernism (e.g. the Polish sociologist Zygmunt Bauman is not only quoted there a number of times, but he is also analysed in detail). However, one would not find there the names so often quoted nowadays anywhere, like Jacques Derrida, Noam Chomsky or Leszek Kołakowski, but Claude Levi Strauss, Karl Marx, Richard Rorty and Karl Popper found their way into it. Author of one of the papers says for that instance, would he happen to be believer of the latter, he would create a new ontological category of the "fourth world", populated by new entities (computers), understood as "human created self-operating artifacts".

Of how wide the scope of interest of the authors is, or – on the other hand – of how desperate they are in their search for an answer to the question raised, one can learn from that they reach even to the results of research once carried by Bronisław Malinowski among local population on Triobriand Islands [Westrup_2002].

While one of the authors attempts to relate the human versus computer system (understood as hardware, and software operating on it) to the interrelation between actor (always human) and agent (a device) [Floyd_2002], the other calls for the theory of networks of actors (derived from anthropology), where both – a human and a machine can equally be actors [Klischewski_2002]. That does not mean research of those two authors goes into opposite directions, and the results of the investigation of each of them are contradictory. That only means their research apparatus used to investigate the same subject is basically different.

The relation between the concept of actors, and the quality of software is presented in table 1.

Table 1. The quality of software and the concept of actors.

Aspect of software quality	Area of concern	Key actors
Profitable	Business	Company
Used	Competition	Company and competitors
Useful	Functionality	Users
Usable	Interface design	Users
Functional computer program	Programming	Software engineers

Source: [Kaptelinin_2002]

If one would decide to find any common motives among papers presented there, that would without doubt be the methodology of "*participatory design*", aiming at the creation of IT systems with the active (participatory) presence of their end-users, by developing consecutive prototypes, bringing, with each iteration, the final shape of the future system closer and closer.

This concept in fact isn't anything new, but it is performed using the software tools available nowadays.

An even more sophisticated method of participatory design, also quoted many times in the book is the "*computer aided co-operation*" [Norbjerg_2002]. One can only guess that the authors do mean in this case the various methods of computer assisted system creation, with significant participation of their future users. Unfortunately, only to guess, because the most detail definition of that approach given in the paper is of "*those, who owned the knowledge (about the subject – B.P.), and those who do the real design*", while no party of that relation is ready for mutual co-operation. In practice however, those methodologies, despite the comprehensive advertising during eighties, never became the main way the IT systems are created, simply because those systems did not meet the real needs of users [Westrup_2002].

Many papers populating the book discussed here call for the theory of Alexi A. Leontiev. It was Leontiev, who along with two other Russian sociologists (Lev S. Vygotsky and Sergei L. Rubinstein) worked on his theory since the twenties of the twentieth century. In contradiction to the purely biological theories of evolution, Leontiev sees in its process a special role of a human mind, the development of which resulted from continuous and successful adaptation of humans to their gradually complicating environment. According to Leontiev – there is no way to explain the development leap of human mind, using the biological theory of evolution only. This is simply because there were no significant biological channels to human organism during the evolution period. The consequence of that (and the main factor contributing to the problems raised in the book), are the creative abilities of humans, which still do overcome those of computers by many orders of magnitude.

Other paper contained in the book concentrates on the research of the working place environment, which could bring some kind of breakthrough solution to the area of our interest. However the two methodologies resulting from that (one of these is the widely known UML, the other – so called *Communicating Sequential Processes* – CSP), are far from expectations placed with them, and are not capable to reflect the level of complexity experienced in practice.

Another paper comes up with a promising methodology of "*Locales Framework*" [Fitzpatrick_2002]. However having analysed it closer, one can once again come to the obvious conclusion, that there is a paramount need to understand and comprehend the existing state first, and then to define what is actually required. Having done that it would be enough to bridge one with the other, and achieve the required target in this way. Unfortunately, the methodology in question does not give any indication of how to do it in practical terms. This weakness is clearly visible even to the author herself. At one point she says that the success of that methodology, (which is algorithmic by assumption) depends on "*meaningful effort, to interpret, expand, widen and – if needed – eliminate consecutive aspects of place of action*". In such a complex context – only the last of the aspects named ("to eliminate") seems to be encouragingly interesting. Especially, when another author of another paper introduces exactly the "*method of elimination*", by which he means the elimination from analysis of those data and events, which, however objectively present, do not fit the model adopted in advance.

That leads to two main conclusions:

- One can not assume in advance, that every unit attempting to create some software (and it is irrelevant whether this unit is a separate company, or a part of some bigger organisation), will guide itself with its customers interests in mind,
- The requirements put against an IT system result from a process of negotiations, and not from more or less detail inventory of what is actually required.

If one agrees with the above, then out of the three ways named in the book, in which the social sciences can positively affect the practice of software creation (1 - finding better ways to understand users; 2 - widening the way, the software creators do see how humans are using technology; 3 - change of the values underlying the basics of software creation), only the last of them seems to be of real importance.

The book analysed here comprises papers written during the last three years. Despite this (and what disappoints the reader to some extent), none of the papers discusses the creation of IT systems available over the Internet. Internet systems are especially suitable for the kind of research, since they would allow for interrogation of a kind of user, which seems to be ideal for that kind of research. A user, which may be an average member of the general public, what is more than a person representing a relatively small group of professional individuals, interested only in the systems designed to service the selected, narrow in this context, named areas.

Most of the Internet systems currently available over the Net (including numerous web pages), fail to be “socially related” (in the context of this paper), and – as a consequence – fail to meet the basic and obvious expectations of their users.

Such a bulky work like [Diettrich_2002] could not go without examples of major failures from the area discussed. These examples of failures comprise, in a self-critical manner, a number of experimental initiatives undertaken by the authors of selected papers themselves, and the undertakings, which, for various reasons, have never reached completion.

As an example of this the book quotes the London Stock Exchange project Taurus, which had been abandoned in 1993, after spending almost $700 million, without even single module of the system put into operation. There were numerous reasons for that, and many of them were never made clear [Bergman_2002].

It is a pity the authors did not reach for one of the three biggest ever IT project failures (Ariane 5 space rocket flight control system [Ariane_1996, Baber_1997], London Ambulance Service ambulance despatching services system [Allen_1994], and Therac-25 radiotherapy control software [Neuman_1998, Warboys_1995]). All those disasters resulted purely and entirely from failures introduced by IT design and programming.

4. Conclusion

The [Diettrich_2002] collection of papers constitutes and interesting reading to both – an IT specialist interested in mastering his/her professional skills, and also to somebody interested more in the social aspects of software interfacing with its users.

The book as a whole leaves the reader somehow disappointed, since it does not bring the clear and expected answer to the questions raised.

The book however allows for a broader view on how complicated and sensitive the subject presented in it could be. The serious doubt remains with the reader, whether the papers collected in the book have brought the solution of the problem considered any nearer to resolving. That does not mean the problem raised there is of no importance, or that the search for its solution should be abandoned for the simple reason we're unable to see it or to name it right now.

In fact and to the contrary – such a research is more than needed and welcomed. It is equally needed by the two parties participating in fact in a single task –by the members of the IT community on one hand, and by future IT users on the other.

Literature:

[Allen_1994] Allen, Gary, *Report Details London Ambulance Service Problems*, Dispatch Services Inc., 1994

[Anderson_1997] Anderson, Bob, *Work, Ethnography and System Design,* in: Kent, A., Williams, R.J., (eds.), The Encyklopedia of Microcomputers, vol. 20, New York, 1997

[Ariane_1996] *ARIANE-5, Flight 501 Failure*, Report by the Inquiry Board, Paris, 1996

[Baber_1997] Baber, Robert Laurance, *The Ariane 5 explosion as seen by a software engineer,* University of Witwatersrand - Computer Science Department, http://www.cs.wits.ac.za/~bob/ariane5.htm, 1997

[Bergman_2002] Bergman, Mark, King, John Leslie, Lyytinen, Kalle, *Large-Scale Requirements Analysis as Heterogeneous Engineering,* in: Dittrich Yvonne, Floyd, Christiane, Klischewski, Ralf, (editors) Social Thinking - Software Practice, Mit pres, Cambridge, 2002

[Carr_2003] Carr, Nicolas G., *IT Doesn't Matter,* Harvard Business Review, May 2003

[Fitzpatrick_2002] Fitzpatrick, Geraldine, *The Locales Framework: Making Social Thinking Accessible for Software Practitioners*, in: Dittrich Yvonne, Floyd, Christiane, Klischewski, Ralf, (editors) Social Thinking - Software Practice, Mit pres, Cambridge, 2002

[Guess_2001] Guess, Frank M., Improving Information Quality And Information Technology Systems In The 21st Century, University of Tennessee, Knoxville, 2001

[Jirotka_2002] Jirotka, Marina, Luff, Paul, Representing and Modelling Collaborative Practices for Systems Development, in: Dittrich Yvonne, Floyd, Christiane, Klischewski, Ralf, (editors) Social Thinking - Software Practice, Mit pres, Cambridge, 2002

[Kaptelinin_2002] Kaptelinin, Victor, Making Use of Social Thinking: The Challenge of Bridging Activity Systems, in: Dittrich Yvonne, Floyd, Christiane, Klischewski, Ralf, (editors) Social Thinking - Software Practice, Mit pres, Cambridge, 2002

[Kopetz_1976] Kopetz, Hermann, Softwarezuverlaessigkeit, Carl Hanser Verlag, Muenchen, 1976

[Meintrup_2002] Meintrup, David, Codierungstheorie, Uniwersitaet det Bundeswehr, Muenchen, 2002

[Myers_1980] Myers, Glenford J., Projektowanie niezawodnego oprogramowania, WNT, Warszawa, 1980

[Neuman_1998] Neumann, Peter G., *Illustrative Risks to the Public in the Use of Computer Systems and Related Technology,* Software Risk Institute International, Menlo Park, 1998

[Oberquelle_2002] Oberquelle, Horst, Useware Design and Evolution: Bridging Social Thinking and Software Construction, in: Dittrich Yvonne, Floyd, Christiane, Klischewski, Ralf, (editors) Social Thinking - Software Practice, Mit pres, Cambridge, 2002

[Paulk_1993] Paulk, Mark C., Curtis, Bill, Chrissis, Mary Beth, Weber, Chrles V., The Capability Maturity Model for Software, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1993

[Robertsons_1999] Robertson, Suzanne and James, Mastering the requirements process, Addison-Wesley, London, 1999

[Warboys_1995] Warboys, Brian, *The Software Paradigm*, Ingenuity ICL Technical Journal, vol 10, issue 1, May 1995

[Westrup_2002] Westrup, Chris, On Retrieving Skilled Practices: The Contribution of Ethnography to Software Development, in: Dittrich Yvonne, Floyd, Christiane, Klischewski, Ralf, (editors) Social Thinking - Software Practice, Mit pres, Cambridge, 2002