Making Users the Subject of a Software Process to Build High Quality Software for Public Organizations

Barbara Begier
Poznan University of Technology
Barbara.Begier@put.poznan.pl

Abstract

The UID (*User-Involved Development*) approach, developed to improve quality of software for public organizations, is outlined in reference to several attempts to involve users in a software process. Its assumptions are presented to provide continual feedback between software developers and users, and to establish the process of software product assessment. Examples of quality measures are listed. Examples of the required personal competencies are given, too.

1. Introduction

Information technologies look promising to improve general functions and particular activities in public organizations. Unfortunately, the percentage of successful software products applied in public administration, hospitals, social insurances, judiciary, emergency services, etc., is shockingly low [BrCS 2004]. The methodologies, developed to produce software for military applications, control devices, and mobile communication, don't work in the case of software systems for public organizations.

In the today's economy the high productivity is the main criterion of success. This statement refers also to a ***software process***. Hard methodologies, focused on making software process fast and repeatable, dominate on the market. Software solutions follow the developers' point of view because of their isolation from software users.

The high productivity results often in poor quality of software products built under the pressure of time instead of quality features. So far it is a regular case that an inquirer may apply for something immediately but he/she receives the proper answer or correct document in three months (quoted examples come from Poland and France). Reasons underlying wrong solutions are based on a concentration on particular functions instead of the complete business technology. Software developers make a wrong assumption that end users are a homogenous group of people who work always perfectly. Thus a lot of failures

arise because of the mistaken data and improper behaviours of users. Insufficient efforts have been made to detect and to correct mistaken data. Too optimistic estimations concern migrations and transfers of data. Yesterday's solutions are transferred into an information system intended to support specified *processes* operating on correctly digitized data.

The aim is to improve software production for governmental and other public organizations. This kind of software is built by making use of public funds. The change of technology in the public organization should be oriented to its ***mission*** which is superior to profitability of software. This mission is to provide and improve specified services for citizens who are indirect users. The software system should be designed along with a technology of its every day usage.

The ***level of users' satisfaction from a software product*** is assumed the ***main measure of software quality*** in the presented UID (*User-Involved Development*) approach. The goal is to raise the level of average user's satisfaction. To improve software quality the direct and indirect users are supposed to cooperate with designers in the software process. So called soft methodologies associated with users' involvement have been developed since the early seventies but their share in software production for public organizations still remains slight.

Users should leave peripheries of the software process and ***become its subject***. Users take part in the specified primary and supporting processes. The idea of the UID derives partially from author's experience with poor software quality. This approach is also a result of the research concerning software quality, including quality assessment − many of the polled respondents show their willingness to influence on a software product [Begier 2003a]. An essential feature of the UID is the permanent feedback from users on a software product. The UID approach has been applied at first to the evolutional development of software supporting specified calculations in civil engineering. After four iterations of the collaborative development including periodical assessment of software quality [Begier 2003b] the level of users' satisfaction from a product is much better than that at the beginning. Despite all differences between the software applied by civil engineers and that used in public organizations ***the idea to incorporate direct and also indirect users*** in the software process is worth working out. The quality aspects are mainly focused on ***usability*** of a product from the users' point of view. Some examples of quality measures are given in the section 6.

Another idea is to extend the ***competency management*** in software companies − competencies which enable cooperation with users should be developed. Users differ from developers in age, gender, skills, and personalities. Author's research in four organizations showed that 56 % of users were females in the 40 to 50 age bracket [Begier 2003a], while software factories are dominated by men under 35. Since program authors become currently rather creators of software models than programmers, then a set of their required competencies should change to include also competencies which enable to cooperate with people of various profession, age, culture, and gender. This statement, not new in other fields, is still passed over in computing faculties.

2. Attempts to fill a gap between hopes and a reality of software solutions

Many papers concerning ***software quality*** have been published. Recommended methods are based on the right assumption that the good process results in the good software product. Most of them refer to the ISO 9001 [ISO 2000] and/or to the Capability Maturity Model (CMM) [Paulk 1995]. But the problem of quality is still open. Is it enough to meet standards to achieve a good process? Whose quality criteria are considered? How to improve a software process to provide the quality expected by users?

The applied practice is to specify requirements, including quality attributes, and then to build a software product. Developers are convinced that it is possible to get the detailed requirements at the beginning of the product life cycle and that it is sufficient to provide software quality. In the real life, after the first version is installed, the never ending improvements are added to the software system.

Failures in software production question a credibility of classical phases of the software development [Ambler] and cause to look for another approaches. Instead of rigorous methods to specify required functions and other features of a future product at the very beginning of a software process (*Feature driven development*), the ***agile methods*** have been developed to modify the previously applied methodologies [Agile FDD 2004] by emphasizing interactions and customer collaboration.

The idea to involve users in the software process has been developed since the early seventies, starting from the Checkland's concept of soft methodologies [Checkland 1999], and the Participatory Design suggested by Nygaard [Winograd 1996]. Supporters of various agile methods [AgileM 2005] emphasize the ecology approach [Arbaoui 1999] focused on human aspects of the software process. They share the principles expressed in the Agile Manifesto [PriAM 2001] − one of them states that *the most efficient and effective method of conveying information to and within a team is face-to-face conversation*. From among agile approaches the XP methodology [Beck] has recently gained a great popularity in telecommunication and aircraft industry.

The Socio-technical design [Mumford 1983] and the Value Sensitive Design [Friedman 1997] is based on a looking for optimal solutions from a social point of view. A satisfaction of the employee from his/her work is an important value in these approaches. But people factors in software production are still reduced to the management of human resources at the developer's side [DeMarco 1987, Curtis 2001, Turner 2003].

The main idea of the User-centered design of web applications [Katz 1998] is to involve users to provide an ease of use focused on a friendly interface of the software product. The IBM User-Centered Design incorporated in the software process [UCD IBM 1999] requires to appoint interdisciplinary teams who use techniques applied in marketing methods to define the target audience of the created internet application, then to build a list of their preferences, etc. The general call for a user-driven development process in web time [Cloyd 2000] has

appeared to respect human factors of the software process oriented on e-business and Web applications. It's been again focused on user interface and usability tests.

Specification of *use cases* expressed in the UML notation [UML] and/or a creation of *prioritized and refined scenarios* [Barbacci 2003] bring positive results. ***Stakeholders*** are individuals affected somehow by a software product, like end users, installers, administrators, system architects, software engineers and designers. System stakeholders are engaged early in the life cycle to discover quality attributes and to develop possible scenarios.

An *emphasis on a usability* of a software product is based on an assumption that any tool of work should improve its productivity and provide its ease. If new tools eliminate previous problems and a discomfort at work by reducing stress then employee's satisfaction grows − this is the most important criterion of quality of life. Principles and general instructions of the *user-centred approach for interactive systems* have been given in the INUSE project [INUSE 2002]. Software development is a collaborative process which requires *multi-disciplinary teams* − two groups of specialists are involved in a process: software designers and software users. They are supposed together to specify and design proper solutions, then to test and discuss prototypes.

Since the general intellectual level of software developers is estimated as the near or even under an average in a society because of the shortage of talents [Curtis 2001] thus the quality *experts* are expected to play their great role in providing software quality. Experts are facilitators who ask stakeholders for specification and correction of basic scenarios. Experts also play roles of inspectors involved in reviews and inspections in the software development cycle. Unfortunately, they slightly know the real life problems which may appear during an exploitation of the given product. The developed scenarios usually concern the typical business matters involving basic system functions. No indirect users are involved in the process.

3. Software systems developed for public organizations

There is no one kind of users in the case of a software system developed for a public organization. The *customer* or a *purchaser* is an organization (for example: a hospital, a ministry, the Municipal Council, the tax office), which orders the specified software product. The term *client* refers to the organization in which a product is installed. The words *personnel* and *employees* are synonyms in this paper. The *direct user* means any clerk including a person holding a managerial position who uses the considered software system at his/her work. The *indirect user* depends indirectly on the system − he/she is a patient or supplicant of the given organization, like a hospital or register office. The term of *users' representatives* means a set of people who represent direct and indirect

users of any kind − average people, as they really are. Since users are various people thus the users' representatives should be *representative* for the broad spectrum of users.

Software systems developed for public organizations have the following features in common:

- Relatively long life cycle of large software systems. Data are maintained longer than programs.
- That's *a priori* assumed that system development is carried out step by step in several years.
- Evolutional development of the system takes place till it's withdraw. Changes of its functionality have to keep up with the dynamically changing legislative rules, an organizational progress and new technologies.
- Additional source of new requirements comes from users' attitudes and their personal development, according to the hierarchy of needs described by Maslow [Maslow 1954] − users propose new requirements after their basic needs are satisfied.
- Just clerks of the given organization but not computer programmers are the direct users.
- Applied software solutions have a great influence on a scope, way, and a quality level of services for customers of the organization. The declared deadlines of offered them services should be met.
- System can be decomposed according to branch and competency structures maintained in the given organization. Generated documents also refer to those structures.
- Software system maintains databases and document bases which should be provided with safety rules as the recommended in the ISO standard 17799.
- Software applications support an interpersonal communication and coordination of works among clerks. It leads step by step to the future automation of workflows.

Software systems for public organizations are built by making use of public funds. An assessment of software systems is made potentially by all taxpayers and has some indirect influence on social relationships and feelings of citizens. One may expect that investments in software systems will be better supervised by social representatives in the future than it takes place nowadays.

4. Software users actively involved in a software process

Software developers seem so far to show their prejudice against less qualified, in their opinion, users. A reluctance to get involved in cooperation with software users also arises due to the general criticism of unsatisfying and unreliable software products. In turn, the users' aversion to software developers is a result of low quality of software products and an arrogance of its authors.

The depicted above attitudes may change diametrically. Factors conditioning that change are like:

- Software authors do not test and assess software products only by themselves.
- Users are allowed and asked to assess software products applied in public organizations.
- Direct users are no more supposed to adapt themselves to the software system. They are not an addition to the system or perfect robots. They play an active role in the software process.
- Software product is adapted to various users' needs and capabilities.
- The cooperation of developers and users is not limited to the requirements phase.
- It's been assumed that a scope of requirements will grow each time after the new version of the software product is introduced.
- The representatives of indirect users also take part in the software process. Methods practiced in sociology are applied to choose the representative sample of indirect users to involve them in interviews, meetings, joint reviews, questionnaires, and another forms of software assessment. Thus the quality level of services provided for citizens by a public organization using the software product is indirectly an object of the assessment.
- Software users are prepared and skilled enough to take part in specified forms of cooperation with software developers in the software process.
- Software companies are financially dependent of users' assessments of the delivered products.
- Technical competencies of employees in software factories are not the only ones or the most important. Personality-based competencies essential to cooperate with users' representatives play an important role in the competency management.
- Direct and indirect users are conscious of their rights to assess software products and to have their influence on quality features and forms of the product.

To sum up, users become the ***subject*** of the software process. At the first glance, users' capabilities to be partners of software developers from the very beginning of the software process may seem problematic. But the estimated level of users' skills in information technologies is based on the yesterday's assessment and results. It evolves as an element of the educational boom. Present pupils of primary schools use computers as common tools of work and play. They will have no barriers to use computers at their future work and also to participate in software process in a professional manner. They will not tolerate the today's inconveniences of using imperfect software products and not accept delays in delivery of required software features. Besides these arguments it's been assumed that the role of users is more and more active in the subsequent software development cycles.

5. Software life cycle in the UID approach

The UID (*User-Involved Development*) approach has been introduced and recommended [Begier 2005] to be applied in software development for public organizations. Expanding an idea to involve users in the software process, it is oriented to ***direct and also to indirect users*** of a software product. Their expectations, together with all other specified requirements, are considered in the whole life cycle. So users' involvement is not limited to the requirement specification. At the very beginning of a software development a *questionnaire survey of possibly all users* is recommended to get learn who users are. An assessment of a product by a broad spectrum of users takes place each time the new version is introduced. The model of the software life cycle in the UID approach is shown in a Figure 1.

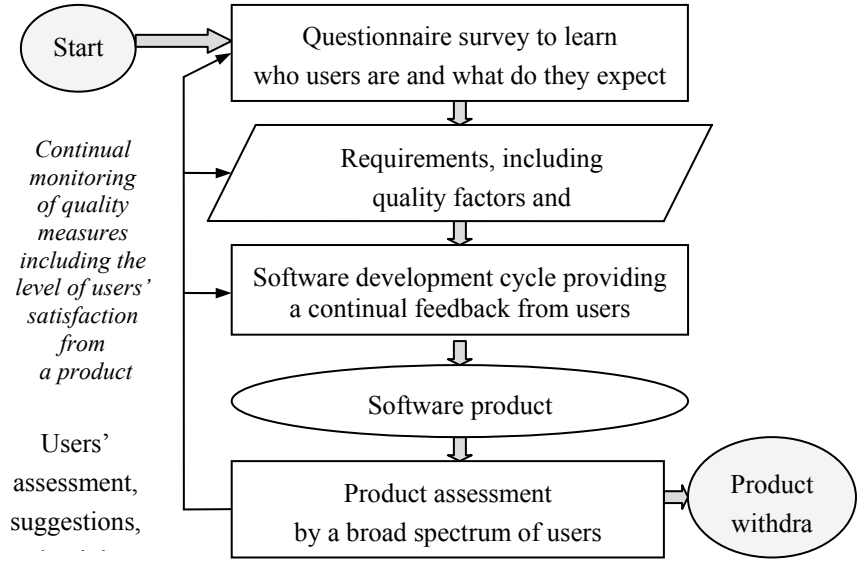

Figure 1. The software life cycle in the UID approach.

In general, there are several loops of continual feedback between developers and users to provide that users' opinions are respected and considered on line. Initial phases of software development cycle are focused on usage modelling and support of business processes. The most external loop refers to an assessment of the product, delivered and installed at the purchaser's side. The experience shows that some users give not only their notes to the specified features of the software product but also include their constructive remarks in questionnaires [Begier 2003a, Begier 2003b].

The core feature of the UID is to emphasize the role of direct and also indirect users in software assessment. Results of the software assessment are an input of

the next development cycle as illustrated in the Figure 2. Users enable to detect wrong solutions or lacks in software functionality by reporting problems which occurred at their every day work. Users' presence makes available to prioritize software functions and other requirements.

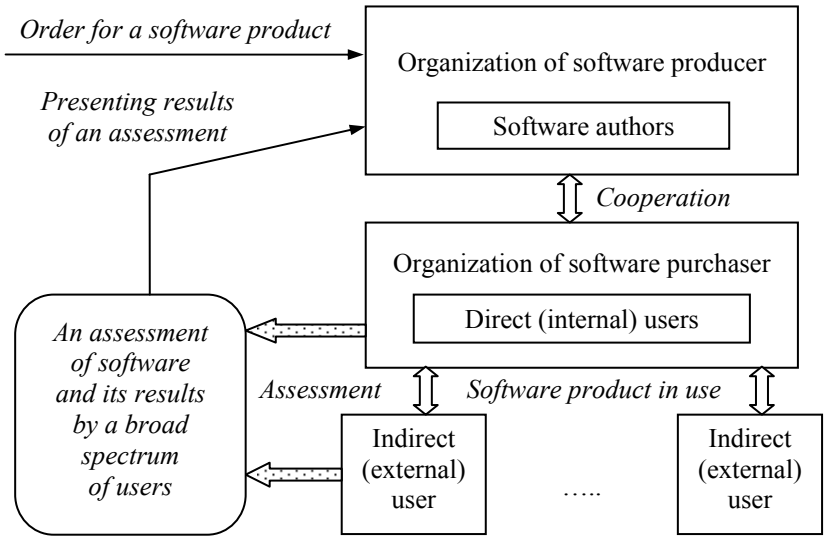

Figure 2. Permanent feedback between software authors and users.

Domain and quality experts are also involved in the process of quality assessment but their presence is not equivalent to users' involvement in the software process. Experts and users are different groups of people and show different competencies − experts trace a conformity with standards and similar solutions while users assess the usability of a product and its usefulness to manage regular and also untypical tasks. Indirect users assess a *level of services* provided by a given organization after its informatization.

6. Processes and quality measures

The UID approach requires establishing an initial questionnaire survey to learn who users are. Various processes of cooperation with users are defined and established, including work in joint teams, brain storm sessions, meetings half-and-half, business process modelling and analysis, software product assessment by its direct and indirect users. Users also take part in the specified, previously performed without their presence, processes − they participate in test planning and in specifying control lists for reviews and inspections.

Close relationships with users generate new requirements concerning the supporting infrastructure. The extended CRM (*Customer Relationship*

Management) system is required to maintain data concerning joint activities and involved users, like delegated members of teams, participants of meetings and brain storm sessions, persons responsible for particular documents, participants of joint reviews, authors of test cases and scenarios, co-authors of check lists developed for software inspections, users taking part in an assessment of a product, etc. The prototype of such CRM system has been developed and described in [Begier 2004].

Quality criteria and measures are partly derived from the ISO 9126 standard. They are related to business goals and should also meet social needs and ethical principles. To emphasize the safety it's been separated from functionality. The recommended here measures of software quality (ca 100 measures in one questionnaire of quality assessment) are divided into the following groups:

1. General usefulness for an organization
2. Functionality
3. Safety
4. Ease of use
5. Efficiency of customers' service
6. Provided data verification and correction
7. Users' impressions and feelings
8. Reliability
9. Ethical point of view and social expectations
10. Portability

Examples of quality measures, specified from an indirect user's point of view, in the fifth group are the following:

5.1. An average time of each specified kind of service for an inquirer's application.
5.2. The real time of data transfer from system server.
5.3. The level specifying how much activities of data acquisition and verification are time-consuming.
5.4. The total time required to fix something for the supplicant (the times of all performed activities), for example to get an excerpt (after a specified modification) from the land register.
5.5. The worst time to get the system correct response for each specified service (in the case of mistaken or incomplete data, changing personnel not aware of what's been done, etc.), for example, to get the required certificate of being able to marry from the register office.
5.6. Real number of visits in the organization to deal with the specified business matter, for example, to get an updated excerpt from the land records.
5.7. Number of required documents and data (forms, attachments, spaces provided, etc.) which should be presented each time by an applicant (although those data are maintained in the software system).
5.8. Level of the provided possibility to correct mistaken data immediately.
5.9. Number of words required to correct mistaken data.
5.10. Number of iterations required in the case of wrong service or mistaken data.

5.11. The ratio of the time required for service including data correction to the time of a traditional service.
5.12. Number of mistaken or incomplete data met in one day of work (an average monthly).
5.13. Number of communication errors daily.
5.14. Number of net problems (suspensions, no response, etc.) monthly.
5.15. Paper savings (the number of currently generated document pages compared with the previously printed number of pages in the same case).
5.16. Number of cases which took more than one day in service.
5.17. Number of cases which took more than one month to complete their service.

The expected forms of answers are precisely specified for each metric to avoid misunderstandings. Direct users give their notes to all specified measures. The quoted above measures in the fifth group are also assessed by representatives of indirect users who may also assess the ninth subset of metrics.

After the specified features are measured, a process of their refinement should be initialized if necessary. The refinement specification contains: references to the specified scenario, generated artefacts, given measures, described context, system response, unsatisfied quality attributes, sources of wrong service, their possible reasons, reported problems, possible issues and proposals of improvement. The product refinement also involves the process improvement to avoid similar inadvertences in the next iteration of system development.

7. Expected competencies and preferred roles of software developers

Software developers recruit themselves from among graduates of computer science, software engineering, and similar faculties − today's students will be authors of software systems in the near future. Software companies should recruit people able to work in teams, provide quality, and cooperate with users. Following this line of thought the questionnaire survey, concerning competencies required for possible professional roles in computing, was carried out in December 2005. Just personality features were the objects of research. An initial set of competencies has been specified separately for each of eight professional roles assumed in a software company: *analyst*, *software designer*, *team leader*, *manager*, *quality engineer*, *programmer/tester*, *service worker*, *trainer in computing*.

Participants of this survey were students of the diploma semester in the field of computer science in Poznan University of Technology. They have not been taught or trained before in competency management [Wood 1998]. Respondents had to give notes (using the scale <1, 5>) to competencies, listed in an alphabetical order for particular professional roles and possibly to add more competencies, important from their points of view. At the end of the survey

respondents were asked to declare which professional role they prefer to perform by themselves being best prepared during their studies to perform it. Some students work in software companies as an auxiliary staff so the results of the survey show not only students' imagination.

As results, the four top competencies of an *analyst* are: analytical thinking, accuracy, inquisitiveness, and a capability to accept someone else's point of view. Only 8 per cent of those polled point out a competency of logical thought. A *software designer* should show subsequently: general creativity, capability to create new solutions, capability to accept someone else's point of view, goal orientation, inquisitiveness, and self-criticism. Respondents value as additional competencies: optimism, patience, and resistance to stress. The *service worker* is the only one in students' opinion who should be able to cooperate with other people and to demonstrate a personal culture and politeness. It seems that employees who perform any other roles do not leave their office and do not speak with users. Students work individually − in their opinion only a *team leader* is supposed to show the leadership, achievement orientation, and also capability to listen to others, to solve conflicts, and to tolerate different attitudes or behaviours. The *quality engineer* first of all should be: responsible, able to a reliable assessment, and self-reliant. The competencies of assertiveness and perceptiveness have been added to the initially specified list.

Sets of competencies constitute the required competency profiles considered in the recruitment process in a software organization. Similar procedures may be applied to select appropriate persons to play the required roles in team.

The survey has illustrated what competencies, in students' opinion, are required in software company and also what competencies have been developed by their own. It has revealed that 30 per cent of students prefer the role of an analyst and the next 19 % would like to work as the programmer/tester. No one of respondents is going to be a quality engineer. Simply the university education does not prepare students to provide quality of software products.

8. Final remarks

Users' involvement in software development does not deny an idea of automation of the software process. The current trends in software engineering, like following the concept of Model Driven Architecture (MDA), including business modelling, creating UML models, and then updating and exchanging models, and so on, do not exclude a presence of users who may participate in software development and its product assessment.

The proper solutions of business processes supported by the software system are the source of users' satisfaction. The question is a cost to keep users involved in a software life cycle. A high level of usability translates into high productivity. Due to the IBM experience every dollar invested in the ease of use returns $10 to $100 [UCD IBM 1999] because of the less number of failures and required modifications. Satisfied users make all business efficient and effective −

customers keep cooperating with a software producer which provides collaboration resulting in the high quality products.

9. References

[Arbaoui 1999] Arbaoui S., Lonchamp J., Montangero C., The Human Dimension of the Software Process, Chapter 7 in: *Software Process: Principles, Methodology, Technology* (1999), pp. 165−200.

[AgileM 2005] *Agile Modeling(AM) Home Page*, http://www.agilemodeling.com/, 2001−2005.

[Agile FDD 2004] *Agile Software Development using Feature Driven Development (FDD)*, Nebulon Pty. LTD., http://ww.nebulon.com.fdd, 1993−2004.

[Ambler 2005] Ambler S.W., Phases Examined: Why Requirements, Analysis, and Design No Longer Make Sense, http://www.agilemodeling.com/essays/phasesExamined.htm, 2003−2005.

[Barbacci 2003] Barbacci M. R., Ellison R., Lattanze A. J., Stafford J. A., Weinstock Ch. B., Wood W. G., *Quality Attribute Workshops (QAWs), Third Edition*, CMU/SEI-2003-TR-016, Software Engineering Institute, Pittsburgh (USA), August 2003.

[Beck 2000] Beck K., *Extreme Programming Explained*, Addison-Wesley, Upper Saddle River, 2000.

[Begier 2003a] Begier B., Software quality assessment by users, Chapter XXVII in: *Problems and methods of software engineering*, Wydawnictwa Naukowo-Techniczne 2003, pp. 417−431.

[Begier 2003b] Begier B., Wdowicki J., Quality assessment of software applied in civil engineering by the users, *4th National Conference on Methods and Computer Systems in Research and Engineering MSK'03*, Kraków, November 26−28, 2003, pp. 47−552.

[Begier 2004] Begier B., Customer relationship management in software company, Chapter XVI in: *Software engineering. New challenge*, WNT, Warszawa 2004, pp. 213−226.

[Begier 2005] Begier B., The UID Approach – the Balance between Hard and Soft Methodologies, In: *Software Engineering: Evolution and Emerging Technologies*, IOS Press, Amsterdam, 2005, pp. 15−26.

[BrCS 2004] British Computer Society: The challenges of complex IT products. The report of a working group from The Royal Academy of Engineering and the British Computer Society, April 2004, http://www.bcs.org/BCS/News/positionsandresponses/positions

[Checkland 1999] Checkland P., *Soft Systems Methodology in Action*, John Wiley & Sons, July 1999.

[Cloyd 2000] Cloyd M. H., Creating a user-driven development process: in web time, *6th Conference on Human Factors & the Web*, Austin, Texas, June 19, 2000.

[Curtis 2001] Curtis B., Hefley W. E., Miller S. A., *People Capability Maturity Model*® (*P-CMM*®) *Version 2.0*, Software Engineering Institute, Pittsburgh (USA), July 2001, http://www.sei.cmm.edu/cmm-p/

[DeMarco 1987] DeMarco T., Lister T., *Peopleware: Productive Projects and Teams*, Dorset House, New York, 1987

[Friedman 1997] Friedman B., *Human Values and the Design of Computer Technology*, Cambridge University Press, 1997

[INUSE 2002] INUSE 6.2, *Handbook of User-Centred Design*, Nectar Project 2002, http://www.ejeisa.com/nectar/inuse/6.2/content.htm

[ISO 2000] International Standard EN ISO 9001: *Quality management systems – Requirements*, International Organization for Standardization, Genève 2000.

[Katz 1998] Katz-Haas R., Ten Guidelines for User-Centered Design, *Usability Interface*, vol. 5, No.1, July 1998, also: www.stcsig.org/usability/newsletter/9807-webguide.html, Society for Technical Communication (STC), 1998−2004.

[Maslow 1954] Maslow A., *Motivation and Personality*, Harper and Row, 1954.

[Mumford 1983] Mumford E., *Designing Human System for New Technology − The ETHICS Method*, 1983, on-line book available at: http://www.enid.u-net.com/C1book1.htm.

[Paulk 1995] Paulk M. C., Weber Ch. V., Chrissis M. B., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, Reading, Massachusetts, 1995.

[PriAM 2001] *Principles behind the Agile Manifesto*, (Kent Beck, Martin Fowler, Jim Highsmith, Robert C. Martin, et al.), http://agilemanifesto.org/principles.html, Agile Alliance, February 2001.

[Turner 2003] Turner R., Boehm B., People Factors in Software Management: Lessons from Comparing Agile and Plan-Driven Methods, *Cross Talk. The Journal of Defence Software Development,* December 2003, pp. 4−8.

[UML] UML 2.0 (Unified Modeling Language). The Current Official Version, OMG UML™ Resource Page, http://www.uml.org/#UML2.0

[UCD IBM 1999] User-Centered Design Process, IBM, http://www-306.ibm.com/ibm/easy/eou_ext.nsf/publish/

[Winograd 1996] Winograd T., *Bringing Design to Software*, Profile 14. Participatory Design, Addison-Wesley 1996.

[Wood 1998] Wood R., Payne T., *Competency-based Recruitment and Selection*, John Wiley & Sons, Chichester (UK), 1998.