In: M. Sikorski & M. Rauterberg (1998) (eds). Proceedings of the International Workshop on Transferring Usability Engineering to Industry (Gdansk, Poland, June 11-14, 1998, pp. 33-37). Gdansk: Technical University of Gdansk

Moderation instead of Modelling: Some Arguments against Formal Engineering Methods

Matthias Rauterberg
Center for Research on User-System Interaction (IPO)
Den Dolech 2, NL-5600 MB Eindhoven, Email: rauterberg@ipo.tue.nl

Abstract. The more formal the used engineering techniques are, the less non-technical facts can be captured. Several business process reengineering and software development projects fail, because the project management concentrates to much on formal methods and modelling approaches. A successful change of work and organisational processes and structures needs primarily the agreement among all involved groups. To come up with a shared understanding, the quality and the quantity of communication are critical success factors. The moderation--instead of modelling--of human activities is of crucial importance. To change an organisation means to persuade people to behave in a different way than before. A model per se does not change anything, only an agreement among all affected parties is the basis of a real change. Participatory design and the adequate moderation of all necessary learning processes are the crucial success factors.

1. Introduction

The more or less formal methodology of 'business process reengineering' (BPR, Hammer and Champy 1993) has been discussed intensively in the last few years. By means of a fundamental redesign of business processes the methodology of BPR wanted to achieve an increase in efficiency of 20 to 40 per cent. As we know today most of all BPR projects in USA failed (Womack 1995). If BPR intends to be more than just 'the latest thing', its fundamentals have to be carefully analysed. A recourse to already known concepts shows, that some connected problems have to be solved to change BPR from 'the latest thing' into real business revolution. What are the reasons for the unsuccessfulness of BPR: Is it the distinction between 'formal' versus 'informal' approaches, or are there other reasons? We will discuss this problem in the context of software development and the implementation of modern technology (e.g., information systems) in enterprises. The most important reason that we can do this, is the fact, that the implementation of modern technology in companies will nearly always affect the work and organisational structure of this company. Software design is work design, and work redesign deals always with humans! This aspect is one of the 'new' insights of Champy (1995).

Analysis of current software development processes brings to light a series of weaknesses and problems, the sources of which lie in the theoretical concepts applied, the traditional procedures followed (especially project management) as well as in the use of inadequate formal design methodologies. DeMarco and Lister (1987) reported that fifty per cent of all investigated projects were stopped, failed, or delivered a product that was never used; the larger the project, the greater the chance to fail! In nearly all of these projects the critical factor for success or failure was *not* the used technology. But, what are the relevant success factors?

A possible answer points to the significance of the 'human factor'. The analysis of actual software development processes shows that there are three essential barriers: the *specification barrier*, the *communication barrier* and the *optimisation barrier*. Speaking quite generally, one of the most important problems lies in coming to a shared understanding by all the affected people of the component of the work system to be automated (Naur 1985)--that is, to find the answers to the questions of 'if', 'where' and 'how' for the planned implementation of technology, to which a shared commitment can be reached. This involves, in particular, determining all the characteristics of the work system that are to be planned anew. Every work system comprises a social and a technical subsystem. An optimal total system must integrate both simultaneously. To arrive at the optimal design for the total enterprise, it is of paramount importance to regard the social subsystem as a system in its own right, endowed with its own specific characteristics and conditions, and a system to be optimised when coupled with the technical subsystem.

2. Barriers in the Framework of Traditional System Development

There is an increasing tendency in information system development to consider not only the functionality that should be automated, but to pay attention to the work and organisational structures around the information system. Simple, intuitive, and powerful techniques are needed to model all relevant parts of the environment in the requirement analysis phase. Contemporary modelling techniques are either (a) weak in expression or (b) cluttered with rigorous details. In case (a) models become too vague to be meaningful, while in case (b) the techniques make modelling of rather simple dynamic systems a complex task and hardly facilitate communication with users.

Today the object-oriented approach is becoming increasingly popular. Object-oriented analysis (OOA), object-oriented-design (OOD) and object-oriented programming (OOP) are well established (Booch 1994, Brunet et al. 1994). Some of the most often stated advantages of OOA are: (a) object-orientation corresponds very well to the human perception of an organisation (it is a 'natural' view), and (b) choosing an object-oriented structure also for the analysis, the system developer achieves a smooth transition to design and implementation. The main critic against advantage (a) is that *processes* are more fundamental than things or objects (e.g. it is difficult to specify declarative business rules within an object-oriented structure), and against (b) that a small gap can be achieved only by pulling analysis to design (this can be done--for example--with a requirement engineering approach that is *problem-* and not target-oriented). BPR projects--with their primary focus on the processes--should be more successful than OOA projects. But this statement seems to be not true. Why?

The *specification barrier* is an immediate problem even at a cursory glance. How can the software developer ascertain that the client is able to specify the requirements for the subsystem to be developed in a complete and accurate way that will not be modified while the project is being carried out? The more formal and detailed the medium used by the client to formulate requirements, the easier it is for the software developer to incorporate these into an appropriate software system. But this presumes that the client has command of a certain amount of expertise. However, the client is not prepared to acquire this--or perhaps is in part not in a position to do so--before the beginning of the software development process. It is therefore necessary to find and implement other ways and means, using from informal through semi-formal to formal specification methods (see Pohl 1993). It would be a grave error with dire consequences to assume that clients--usually people from the middle and upper management level--are able to provide pertinent and adequate information on all requirements for an interactive software system. As a result, the following different perspectives must be taken into consideration in the requirement phases.

The *communications barrier* between applier, user and end-user on the one hand and the software developer on the other is essentially due to the fact that 'technical intelligence' is only inadequately imbedded in the social, historical and [micro] political contexts of technological development. Communication between those involved in the development process can allow non-technical facts to slip through the conceptual net of specialised technical language, which therefore restricts the social character of the technology to the functional and instrumental. The more formal the used techniques are, the less non-technical facts can be captured. The application-oriented jargon of the user flounders on the technical jargon of the developer. This 'gap' can only be bridged to a limited extent by purely linguistic means, because the fact that their respective semantics are conceptually bound makes the ideas applied insufficiently precise. Overcoming this fuzziness requires creating jointly experienced, perceptually shared contexts. Schuler and Namioka (1993) describe several approaches how to do this. Beyond verbal communication, visual means are the ones best suited to this purpose. The stronger the perceptual experience one has of the semantic context of the other, the easier it is to overcome the communications barrier (Ehn and Kyng 1991).

At its best, software development is a procedure for *optimally designing* a product with interactive properties for supporting the performance of work tasks. Because computer science has accumulated quite a treasure-trove of very broadly applicable algorithms, software development is increasingly focusing attention on those facets of application-oriented software which are not amenable to algorithmic treatment. While the purely technical aspects of a software product are best dealt with by optimisation procedures attuned to a technical context, the non-technical context of the application environment aimed at requires the implementation of optimisation procedures of a different nature: moderation of human activities (Wohlgemuth 1995).

It would be false indeed to expect that at the outset of a larger reorganisation of a work system any single group of persons could have a complete, exact and comprehensive view of the ideal for the work system to be set up. Only during the analysis, evaluation and planning processes are the people involvable to develop an increasingly clear picture of what it is that they are really striving for. This is basically why the requirements of the applier seem to 'change'--they do not really change but simply become concrete within the anticipated boundary constraints. This process of *crystallisation* should be allowed to unfold as completely, as pertinently and--from a global perspective--as inexpensively as possible. Completeness can be reached by ensuring that each affected group is involved at least through representatives. Iterative, interactive progress makes the ideal concept increasingly concrete (for more details see Rauterberg, Strohm and Kirsch 1995).

3. First steps to Implement Technology

The analysis phase is frequently the one most neglected (Boehm 1984). This is essentially due to the fact that methods and techniques need to be used primarily the way occupational and organisational sciences have developed and applied them (e.g., Macaulay et al. 1990). Inordinately high costs incur from the troubleshooting required because the analysis was less than optimal (Rauterberg, Strohm and Kirsch 1995). The time has come to engage occupational and organisational consultants at the analysis stage who have been especially trained in *moderation* techniques for software development! Once the analysis of the work system to be optimised has been completed, the next stage is to mould the results obtained into implementable form. Methods of specification with high communicative value are recommended here (Kraut and Streeter 1995), and not a method that enable the project management to retrace a misspecification (the 'traceability problem' see Pohl 1993). Sufficient empirical evidence has accumulated by now to show that task and user oriented procedures in software development not only bring noticeable savings in costs (Rauterberg, Strohm and Kirsch 1995), but also significantly improve the software produced (van Swede and van Vliet 1994). How then, can the barriers mentioned above be overcome?

The first thing is to determine 'if' and 'where' it makes sense to employ technology. "Although the view is still widely held that it is possible to use technology to eliminate the deficiencies of an organisation without questioning the structures of the organisation as a whole, the conclusion is nevertheless usually a false one" (Klotz 1991). The intended division of functions between man and machine is decided during the specification of the tool interface (for a more detailed discussion see Ulich et al. 1991 and Grote et al. 1995). Once those concerned are sufficiently clear about which functions are amenable to automation, the next step which should be taken is to test the screen layout on the end-users with hand-drawn sketches (the extremely inexpensive 'pen and paper' method, see Ehn and Kyng 1991).

The use of prototypes, to illustrate the dynamic and interactive aspects of the tools being developed, is indispensable for specifying the dialogue interface. However, prototypes should only be used very purposefully and selectively to clarify special aspects of the specification--not indiscriminately. Otherwise there looms the inescapable danger of investing too much in the production and maintenance of 'display goods'. A very efficient and inexpensive variation is provided by simulation studies, for example, with the use of hand prepared transparencies, cards, etc. which appear before the user in response to the action taken (Schuler and Namioka 1993).

Simple and fast moderation techniques for involving users include discussion groups with various communication aids (Metaplan, layout sketches, 'screen-dumps', scenarios, etc.), questionnaires for determining the attitudes, opinions and requirements of the users, the 'walk-through' technique for systematically clarifying all possible work steps, as well as targeted interviews aimed at a concrete analysis of the work environment (Macaulay et al. 1990, Nielsen 1993). Very sound simulation methods (e.g. scenarios, 'Wizard of Oz' studies, mock-ups) are available for developing completely new systems without requiring any special hardware or software (Ehn and Kyng 1991). Nielsen (1993) presents a summary of techniques for the analysis and evaluation of interactive computer systems.

4. Conclusion

One of the principal problems of traditional software development lies in the fact that those who have been primarily involved in software development to date have not been willing to recognise that software development is, in most cases, mainly a question of occupational and/or organisational planning and changing. A successful change of work and organisational processes and structures needs primarily the *agreement* among all involved groups. To come up with a shared understanding, the quality and the quantity of *communication* are critical success factors. The moderation--instead of modelling--of human activities is of crucial importance. To change an organisation means to persuade people to behave in a new, different way than before. A model per se does not change anything; only a shared understanding--among all affected parties--is the basis of a real change. Participatory design and the adequate moderation of all necessary learning processes are the crucial success factors. Where information system development is approached from such a perspective, it would be planned from the beginning to engage experts in occupational and organisational planning in the project to *moderate* and *co-ordinate* the whole implementation process (see Kraut and Streeter 1995). This would require interdisciplinary co-operation between occupational and organisational experts on the one hand and software development experts on the other. The extensive qualification required in each of these fields makes it virtually impossible to dispense with such interdisciplinary co-operation. We must start learning to jointly plan technology, organisation and the application of human qualification.

References

Boehm, B., 1984, Verifying and validating software requirements and design specifications. *IEEE Software* 1(1): 75-88.

Booch, G., 1994, Object-oriented analysis and design (Menlo Park: Benjamin/Cummings).

Brunet, J., Cauvet, C., Meddahi, D. and Semmak, F., 1994, Object-oriented analysis in practice. In Advanced information systems engineering by G. Wijers, S. Brinkkemper and T. Wasserman (eds.) (Lecture Notes in Computer Science 811; Berlin: Springer), pp. 293-308.

Champy, J., 1995, Reengineering management: the mandate for a new leadership (New York: Harper & Row).

DeMarco, T. and Lister, T., 1987, Peopleware: productive projects and teams (New York: Dorset).

Ehn, P. and Kyng, M., 1991, Cardboard computers: mocking-it-up or hands-on the future. In Design at Work: Cooperative Design of Computer Systems by J. Greenbaum & M. Kyng (eds.) (Hillsdale: Lawrence Erlbaum), pp. 169–195.

Grote, G., Weik, S., Wäfler, T. and Zölch, M., 1995, Criteria for the complementary allocation of functions in automated work systems and their use in simultaneous engineering projects. *International Journal of Industrial Ergonomics* 16(4-6): 367-382.

Hammer, M. and Champy, J., 1993, Reengineering the corporation : a manifesto for business revolution (New York: Harper Business).

Klotz, U., 1991, Die zweite Ära der Informationstechnik. *Harvard Manager* 13(2): 101-112.

Kraut, E. and Streeter, L., 1995, Coordination in software development. *Communications of the ACM* 38(3): 69-81.

Macaulay, L., Fowler, C., Kirby, M. and Hutt, A., 1990, USTM: a new approach to requirements specification. *Interacting with Computers* 2(1): 92-118.

Naur, P., 1985, Programming as Theory Building. *Microprocessing and Mircoprogramming* 15: 253-261

Nielsen, J., 1993, Usability engineering. London: Academic Press.

Pohl, K., 1993, The three dimensions of requirements engineering. In Advanced information systems engineering by C. Rolland, F. Bodart and C. Cauvet (eds.) (Lecture Notes in Computer Science 685; Berlin: Springer), pp. 275-292.

Rauterberg, M., Strohm, O. and Kirsch, C., 1995, Benefits of user-oriented software development based on an iterative cyclic process model for simultaneous engineering. *International Journal of Industrial Ergonomics* 16(4-6): 391-410.

Schuler, D. and Namioka, A., 1993 (eds.), Participatory design: principles and practices (Hillsdale: Erlbaum).

Ulich, E., Rauterberg, M., Moll, T., Greutmann, T. and Strohm, O., 1991, Task orientation and user-oriented dialogue design. *International Journal of Human Computer Interaction* 3(2): 117-144

van Swede, V. and van Vliet, H., 1994, Consistent development: results of a first study on the relation between project scenario and success. In Advanced information systems engineering by G. Wijers, S. Brinkkemper and T. Wasserman (eds.) (Lecture Notes in Computer Science 811; Berlin: Springer), pp. 80-93.

Wohlgemuth, A.C., 1995 (ed.), Moderation in Organisationen: Problemlösemethode für Führungsleute und Berater (Bern: Haupt).

Womack, J.P., 1995, Neues von Hammer und Champy. *Harvard Business Manager* 18(1): 15-17.