

In: L. Bannon, R. Keil-Slawik & I. Wagner (1995, Eds.), Interdisciplinary foundations of system design and evaluation (Dagstuhl-Seminar-Report, 97, 19.09.-23.09.94 (9438), pp. 42-45). Wadern: Schloss Dagstuhl.

Some remarks about formal and informal specification methods in the context of software development

Matthias Rauterberg
Work and Organisational Psychology Unit
Swiss Federal Institute of Technology (ETH)
Nelkenstrasse 11, CH-8092 Zuerich, rauterberg@rzvax.ethz.ch

Starting point for our reflections

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. This chapter contains an ample store of proposed solutions based on current practice in software development. These point to the significance of a domain specific formalism. Analysis of actual software development processes shows that there are three essential barriers: the specification barrier, the communication barrier and the optimisation barrier (Rauterberg & Strohm 1992).

Speaking quite generally, one of the most important problems lies in coming to a shared understanding by all the affected groups of the component of the worksystem 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 (Rauterberg 1993). Every work system comprises a social and a technical subsystem. An optimal total system must integrate both simultaneously. In order to arrive at the optimal design for the total working system, 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.

Barriers in the framework of traditional software development

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 which 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 measure 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.

It would be a grave error with dire consequences to assume that clients - usually people from the middle and upper echelons of management - 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 analysis and specification phases. The "communications barrier" between applicator, 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 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

In: L. Bannon, R. Keil-Slawik & I. Wagner (1995, Eds.), Interdisciplinary foundations of system design and evaluation (Dagstuhl-Seminar-Report, 97, 19.09.-23.09.94 (9438), pp. 42-45). Wadern: Schloss Dagstuhl.

restricts the social character of the technology to the functional and instrumental.

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 make the ideas applied insufficiently precise. Overcoming this fuzziness requires creating jointly experienced, perceptually shared contexts. 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. 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 focussing attention on those facets of application-oriented software which are unamenable 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. 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. There are methods available for supporting the process of communication which ensure efficient progress (Nielsen 1990) (Nielsen 1993).

First steps to implement technology

The analysis phase is frequently the one most neglected. 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., Maculay et al 1990). Inordinately high costs incur from the troubleshooting required because the analysis was less than optimal (Rauterberg & Strohm 1992). The time has come to engage occupational and organisational consultants at the analysis stage who have been especially trained 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. 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, but also significantly improve the software produced (Karat 1990). How then, can the both 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. The tasks which remain in human hands must have the following characteristics

In: L. Bannon, R. Keil-Slawik & I. Wagner (1995, Eds.), Interdisciplinary foundations of system design and evaluation (Dagstuhl-Seminar-Report, 97, 19.09.-23.09.94 (9438), pp. 42-45). Wadern: Schloss Dagstuhl.

(Volpert 1987) (Ulich et al 1991): 1. sufficient freedom of action and decision-making; 2. adequate time available; 3. sufficient physical activity; 4. concrete contact with material and social conditions at the workplace activities; 5. actual use of a variety of the senses; 6. opportunities for variety when executing tasks; 7. task related communication and immediate interpersonal contact.

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). If the range of templates is very large, then a graphics data bank can be used to manage the templates produced on a graphics editor. The use of prototyping tools is frequently inadvisable, because tool-specific presentation offers a too restrictive range of possibilities. The effect of the structuring measures taken can be assessed with the help of discussion with the end-users, or by means of checklists.

The use of prototypes to illustrate the dynamical 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 (Karat 1990).

Simple and fast 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 (Grudin, Ehrlich & Shriner 1987) (Macaulay et al 1990) (Nielsen 1993). Very sound simulation methods (e.g. scenarios, "Wizard of Oz" studies) are available for developing completely new systems without requiring any special hardware or software. Nielsen (1993) presents a summary of techniques for the analysis and evaluation of interactive computer systems.

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. Were software development to be approached from such a perspective, it would be planned from the beginning to engage experts in occupational and organisational planning in the process of software design. This would require interdisciplinary cooperation 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 cooperation. We must start learning to jointly plan technology, organisation and the application of human qualification.

References

KARAT C-M, 1990: Cost-Benefit Analysis of Iterative Usability Testing. In: DIAPER D et al. (ed.) Human-Computer Interaction - INTERACT '90. Amsterdam: Elsevier Science. 351-356
KLOTZ U, 1991: Die zweite Ära der Informationstechnik. Harvard Manager 13(2):101-112

In: L. Bannon, R. Keil-Slawik & I. Wagner (1995, Eds.), Interdisciplinary foundations of system design and evaluation (Dagstuhl-Seminar-Report, 97, 19.09.-23.09.94 (9438), pp. 42-45). Wadern: Schloss Dagstuhl.

- NAUR P, 1985: Programming as Theory Building. Microprocessing and Mircoprogramming 15: 253-261
- NIELSEN J, 1990: Big paybacks from 'discount' usability engineering. IEEE Software 7(3):107-108
- NIELSEN J, 1993: Usability Engineering. London: Academic Press.
- RAUTERBERG M, 1993: Anforderungen an die Prozessgestaltung der Softwareentwicklung. In: W. Coy, P. Gorny, I. Kopp & C. Skarpelis (Hrsg.) Menschengerechte Software als Wettbewerbsfaktor. (German Chapter of the ACM Berichte, Band 40); Stuttgart: Teubner. 592-599.
- RAUTERBERG M & STROHM O, 1992: Work Organization and Software Development. In: P. Elzer & V. Haase (eds.) Proceedings of 4th IFAC/IFIP Workshop on "Experience with the Management of Software Projects". Annual Review of Automatic Programming 16(2):121-128
- ULICH E, RAUTERBERG M, MOLL T, GREUTMANN T & STROHM O, 1991: Task Orientation and User-Oriented Dialogue Design. International Journal of Human Computer Interaction 3(2):117-144
- VOLPERT W, 1987: Kontrastive Analyse des Verhältnisses von Mensch und Rechner als Grundlage des System-Designs. Zeitschrift für Arbeitswissenschaft 41:147-152