An integral method to make software work

E. Mulder

ErgoS Engineering & Ergonomics, Pb 267, 7500 AG Enschede, The Netherlands

Abstract

A method is proposed to systematically diagnose and improve software in use. The method has an integral evaluation, dealing with the software and the way it is used in its full context. The method leads to integral improvements, not just dealing with software changes, but also with changes in for example work organisation, user instruction or hardware. The characteristics of the method are compared to more common usability evaluations like expert walkthroughs and user tests. Two applications of the method are described: a quick screening and a detailed evaluation. In both cases usability problems are detected in several levels of design, such as functional design (e.g. task flow) and dialogue design (e.g. rearranging and omitting data fields).

Keywords: usability evaluation, human computer interaction, method, software design, ULD/CTD/RSI.

1. Overview

A method is proposed to systematically diagnose and improve software in use. The method in this paper is addressed as 'the *integral* method' to emphasise:

▪ integral evaluation: dealing with the software and the way it is used in its full context;
▪ integral improvements: not just dealing with software changes, but also with changes in for example work organisation.

1.1 History

The integral method has evolved from the way ErgoS contributes to software design projects; where typically the emphasis is on how users fulfil tasks. There is a tradition to consult users at the place of work, carrying out their task.

From the year 2000 on there is an increasing demand to have software evaluated and improved. Sometimes line managers call us: "Will you come over and see? I think we're losing production due to bad software." Other times the initiative comes from ‘Human Resources’ or ‘Health & Safety’: "I wonder whether this software contributes to ULD?" (Upper Limb Disorders, or also referred to as 'RSI' or 'CTD'.)

During the years a method was developed to diagnose and improve software. Beginning 2005 the method was published in [1]; which offers a concise set of software guidelines as well.

1.2 How to make software work better

Working software

Software that works well, facilitates productivity without health risks for the users. The relation between software, productivity and health is described in the Appendix.

To summarise: evaluation and improvement is needed at all design levels of software, among which:

▪ task allocation and task flow;
▪ information design;
▪ dialogue design;
▪ amount and kind of control actions needed.

Integral evaluation

The word 'integral' in the title refers to integral evaluation. The core evaluation is done by a usability expert consulting a user during work. Software is diagnosed together with the way of use in its full context. This differs from most software usability evaluations [2, 3], which separates one or more of: expert, user, task and context.

Integral improvement

The word 'integral' also refers to the integral improvements, not limited to the software itself. In many projects we find that changing the software itself is beyond the budget or beyond the time schedule or simply impossible.

Nevertheless there may be more cost-effective solutions 'outside' the software than inside it. These types of improvement may be found in:

- user instruction on break schedules, efficient actions sequences, keyboard short-cuts, etc.;
- other sources, destinations or formats of data.
- organisation of work: alternative work flows, task enrichment, job rotation, user authorisations, etc.;
- providing different hardware such as displays and input devices.

Changes to software itself also have several levels, including but not limited to (expensive) redesign:

- designing new software;
- selecting alternative software;
- adjusting software in future releases;
- configuration by system administration;
- configuration by user.

1.3 Future developments

The integral method will lead to derived methods, dedicated to certain business domains or types of software. At the moment of writing a method is being developed by ErgoS for the Dutch health insurers; more about this in § 6 Discussion.

ErgoS will do more research on the effectiveness of the method and on the operational usability of ergonomic software guidelines. Unfortunately these guidelines, like a lot of standards, tend to be difficult to use according to their own metrics.

2. The integral method

Step 1. Preparation

In step 1 the project is planned. Decisions are taken about goal, means, time, budget, scope etc. The integral method explicitly demands decisions about which software and which jobs are covered by the evaluation.

Ways to improve software or its use

An important decision in the preparation is about which type of solutions will be dealt with in the project. All the different types mentioned before in 'integral improvement' in § 1.2 may be part of the project, but mostly there are limitations.

Including or excluding certain types of improvement has substantial implications on the criteria used for evaluation and on what deficiencies one should concentrate.

E.g. when software itself may not be changed it is fine to detect awkward dialogue boxes, because there may be ways to avoid them or to reduce their impact. But it is inefficient in this case to focus on potential changes of these dialogue boxes itself.

Step 2. Stakeholders and desk research

The goal of this step is to roughly investigate the main issues like: task flows, complaints of users and line managers, health risks and expected usability issues.

Actions taken are:

- Consulting one or more stakeholders involved in production with this software (line managers).
- Researching materials related to the software like instructions, manuals, types of hardware, screen prints and alike.
- Analysing screen prints to identify whether legibility is an important issue in the project. The integral method possesses an easy technique to gather and analyse screen prints and characters on the level of pixels.

Step 3. Consulting users

This step of user participation may be considered as the core of integral evaluation in the method. The user is not asked to directly pinpoint problems in the software, but he is rather an efficient source for leading the evaluator through the tasks and through the software.

A good and simple question to start off with would be: “Please show me what your most common activity is with this software”. This simple question will be the start of getting insight into:

- the start, flow and end of tasks,
- task frequencies, duration and criticality,
- which information items on the screen are important and frequently read,
- which screens, windows, dialogues, controls give rise to workload (cognitive or physical) and are awkward to use.

Note: besides focussing on the most common user activity it is important to ask for rare but critical activities.

This integral evaluation is described in more detail in § 3, where it is compared to more known usability methods. It is subject to discussion in § 6.

Step 4. Sorting out results and checking with users

Structuring the gathered results

Researching the materials in step 2 and consulting the users in step 3 gives a lot of information which at first has little structure. A first division in the findings can be made by distinguishing the items related to the ‘look & feel’ from the items related to task flow.

The ‘look & feel’ items may be well structured by connecting each item to a type of information or type of control. The task flow items may be structured along the task flow.

Having a meeting with users and stakeholders

It is important to get feedback on the structured results from the users and consulted stakeholders. Presenting the (anonymous) results got so far, to the users and stakeholders in a meeting widens the coverage and makes the results more reliable:

- Users will react on it — often enthusiastically — and thereby give a clear indication whether a resulting item was an accidental problem for a particular user or a structural usability problem.
- Hearing each other, users often come up with more details or examples of usability items.
- Users will help prioritising the items.

Step 5. Getting prioritised items to improve

Assigning priority

The priority of each item is determined by two independent factors:

- quantity (task frequency, duration, criticality) of occurrence of the usability problem;
- quality in negative terms (severity, inconvenience) of the usability problem.

Economic solutions

Besides priority there is another deciding factor whether or not to go for certain improvements: the amount of costs and time needed to implement a solution. E.g.: There may be items with low priority but which are easy to implement. For example distributing a ‘Post-it’ note with the ten most used keyboard short-cuts.

Example of choosing in costs and time limits

Suppose a high priority has been given to the fact that users have to duplicate information by hand from one application to the other (which occurs more often than not).

- A good solution would be integrating the functionality of one application into the other, which probably is very expensive and not to be expected in the next few years.
- A second best solution would be adding an automated data connection; which still may require quite an investment.
- A cost-effective ‘temporal’ solution may be providing keyboard macros or easy buttons on the screen for copying the most wanted information items to the clipboard, which will free the user from laboriously mousing to select and copy these items.

3. Differences with other methods testing usability

3.1 Typical for this integral method

The integral method differs from other evaluation methods in some aspects:

- The integral method has a solid base in the users' expertise executing tasks supported by the software.
- The integral method uses the surplus value which rises from the efficient combination of usability expert and user. The expert lacks task knowledge and has difficulties imagining a users mind. The user lacks discrimination of ineffective interaction and has blind

spots due to being used to the way of working. Both deficiencies will be compensated for in one go.

- The integral method is directed to several types of improvements, not limited to adjusting software.
- The integral method is less effective for designing new software from scratch. There must be access to users performing the proposed tasks.

User + Task + Expert = Integral method

Seen form the perspective of Jakob Nielsen the integral method seems the ultimate evaluation. Nielsen [2] advises to apply alternately the expert review (heuristic) and the user tests in iterative design phases in order to increase the chance to identify usability issues. Bias in [3] proposes a pluralistic usability walkthrough, combining experts and users as well, but this is a group meeting, not dealing with 'real life' task execution.

Production tasks versus public software

To judge about the differences between the integral method and more common methods, one should be aware of the different goals of each.

Notice that the integral method aims to identify the biggest problems in order to make a limited number of leaps towards a more efficient and healthier use of software. The method works best with software in use, for bounded production tasks.

This differs from Nielsen, who aims to identify as many usability problems as possible in order to produce usable new software, often for a large public like web sites.

3.2 Compared to expert walkthroughs

Expert walkthrough (e.g. cognitive walkthrough and heuristic evaluation [2]) is often carried without a working knowledge of tasks.

- This may gives rise to unnecessary work because parts of the interface are evaluated which may hardly be used in practice.
- A frequently occurring mismatch cannot be well evaluated: Is the interaction adequate for the task or does it offer much more (information and controls) than is needed by 95% of the tasks? This mismatch occurs frequently due to the fact that software developers get simple demands like: "design screens which support all these tasks". No one tells them that just 5 of the 20 database fields are involved in 95% of the tasks. In other words when no dedicated screens are designed for the 95% of simple tasks, users are loaded with a far too crowded interface most of the time.
- The expert will miss out on practical task execution, which often differs from what was once specified in the requirements.

3.3 Compared to user tests

Tests carried out by users basically fill the gap caused by the fact that designers are not equal to users and do not perform real life tasks with the software. Therefore user tests are essential in designing software. But this is not enough. Users often do not complain about inefficiencies as long as they understand the system and know how to carry on executing their task.

Example

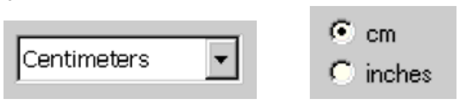

Fig. 1: Dropdown list and radio buttons.

When users have to use the dropdown list in Fig. 1 they know perfectly well what to do: (1) click on the little down arrow, (2) find the value, (3) move the mouse to this value down in the list and (4) click again.

Actually the user may click anywhere in the list control to get it unfolded, but as this does not work with all lists most users tend to just clicking on the "*far too small for frequent clicking*" down-arrow. For frequent use the dropdown list is too laborious.

The radio buttons on the right of Fig. 1 only require one click and that may be anywhere in the imaginary rectangle surrounding button and text label, offering an easier goal for pointing with the mouse.

A user might easily fail to detect that the dropdown list is not efficient and comfortable for very frequent use. Though, an expert in combination with a user will quickly find out whether the control is used frequently and give an adequate priority to get it improved.

(Strangely enough, we quite often see these dropdown lists with just two items: Yes and No; which actually represents the functionality of one simple checkbox.)

4. Application of the integral method

The method was applied to a diversity of applications, like administrative applications, CAD software and intranet portals; all in occupational context with no public use. Applying this method to architectural CAD software has led to a guideline for selecting and configuring CAD software used in building design.

Two examples are described below. The first is a quick screening (about 3 hours per application), the second a detailed evaluation of one application including some redesign. (30 hours).

4. 1 Quick screening of six applications

Rationale

Users complain about small characters, lots of 'mousing' and small screens. The applications are used by about 200 workers and mostly database oriented. Part of the workers deal with clients on the phone.

The goal of the project is to answer these questions:

- Are the applications effective?
- Is there a health risk in using these applications and hardware?
- What type of solutions are there for the user complaints and the problems found in the screening?

Actions and time needed

- step 1 Preparation
  3 hrs: contact by telephone, writing quotation and plan.
- step 2 Stakeholders and desk research
  4 hrs: surveying manuals and reviewing screen prints, identifying issues
- step 3 Consulting users
  4 hrs: six interviews of ½ hr. per application, each time with one or two users.
- step 4
  4 hrs: sorting out results and preparing presentation
  2 hrs: presenting and discussing results so far
- step 5 Getting prioritised items
  2 hrs: adjusting results to get final evaluation report.

Some of the findings & advice

- For a certain critical reporting task every two weeks: cleanup Excel from unused toolbars, borders, etc. to free up space for actual data. Together with a bigger screen, this takes away the need to reduce the zoom factor from 100% to 70%, causing the user to peer at the screen.
- Reduce network reaction times for most tasks by reading data from decentralised (mirror) servers. The few tasks needing to write and update data still use the central server. This improvement aimed to reduce the stress caused by waiting; especially while users had clients on the phone.
- Rearrange data tables to avoid frequent switching to another window to copy one value from it.
- Assemble a task dedicated screen to avoid skipping 10 fields not needed for an ordinary financial booking (40/hour)
- Implement a running total while booking. In the old situation one could only check the correctness of bookings after completing the whole day, hoping no error was made.
- Invest in new screens, slightly bigger and much better focussed.

4.2 Detailed evaluation of one database application

Rationale

A Human Resource Manager expects health risks and workload may be reduced while keeping up or increasing productivity. The application is used by about 100 people with rather monotonous database tasks and dealing with paper forms.

Goal of the projects:

- Lowering mental and physical workload.

Actions and time needed

- step 1 Preparation
  4 hrs: contact by email, writing quotation and plan.
- step 2 Stakeholders and desk research
  6 hrs: visit stakeholders and department using the application, reviewing screen prints, identifying issues.
- step 3 Consulting users
  4 hrs: interviews with three separate users.
- step 4
  4 hrs: sorting out results and preparing presentation
  3 hrs: presenting and discussing results so far
- step 5 Getting prioritised items
  8 hrs: assemble final report, including examples of redesigned screens and alternative task flows.

Some of the findings & advice

- Add dedicated screens for 99% of the tasks, because about 80% of the database fields for these tasks stay empty or hold standard values.

- Rearrange task flows and involved screens in order to facilitate copying data from another application in one go per paper form. In the original situation one had to switch about 2..20 times between two applications for handling one paper form. Users made a solution to this awkward situation by jotting down about 20 values on a bit of paper, which actually was a forbidden practice because of quality regulations.
- Clean up the screens from lines, superfluous labels, borders etc.

5. Conclusions

The method detects usability problems on all design levels of the software, from global task design down to cleaning up screens from graphic frill. The method also leads towards solutions in different domains like work organisation, adjusting software, changing hardware and instructing users.

A more difficult question is: does the method detect all the important usability problems? This cannot be clearly concluded from the applications of the method so far. A positive indication is the fact that users, stakeholders and other usability experts in the feedback meetings hardly ever add new issues to the list of problems found when a detailed evaluation had been done.

6. Discussion

Step 3 (§ 2) is a major component of the integral method. This step efficiently integrates several ways of usability testing in one go. At the same time this pinpoints a weak spot in the method because the evaluator also in one go has to carry out:
- task analysis,
- heuristic evaluation / cognitive walkthrough and
- user test;

all this with guidelines (like ISO 9241) in mind.

To help evaluators in this difficult task it is possible for restricted domains to give them a more restricted framework which is easier to handle. ErgoS is developing this for the Dutch health insurers. This framework helps the evaluator especially in step 3 by supplying a shortlist of about 15 most important and most occurring ergonomic failures in software. Each of these usability items has realistic symptoms described to recognise it and offers indication for improvements.

At the moment of writing it looks like this method and the shortlist of items will facilitate the evaluation and improvement of any administrative, database oriented software. Of course the shortlist of 15 items does not cover all the guidelines.

Appendix: Software, health and productivity

Software influences health (ULD/CTD). Physically this looks quite straight forward. Presumably health risks increase and productivity decreases with:
- more mouse and keyboard actions;
- smaller mouse click areas (demands longer and more precise muscle control);
- unfavourable mouse actions like dragging (pressing a button while moving increases muscle contraction);
- smaller, less legible, characters (tense posture due to peering at the screen).

At the congress (IEA-2006) a special symposium is dedicated to the health risks of computer use: "Unravelling the causes of Upper Extremity Disorders among computer users".

Another major factor is the mental state. It is proven that personality, and stress are important factors in developing ULD/CTD for PC users.

Simple cognitive loads are also important; these increase significantly (co)contraction of muscles, as Van Galen shows in [4]. So for both physical health and mental workload it is important to avoid e.g. memorising and selecting data from crowded screens.

To summarise, software ergonomics is important at all design levels, among which:
- task allocation and task flow;
- information design;
- dialogue design;
- amount and kind of control actions needed.

References

[1] ErgoS, E. (Dutch) Mens Computer Interactie. Arbo-themacahier nr.16, Sdu Uitgevers, Den Haag, The Netherlands, 2005.

[2] Nielsen, J. Usability Engineering. Morgan Kaufmann, San Francisco, 1994.

[3] Bias, R.G. The pluralistic usability walkthrough: Coordinated empathies. In: Nielsen, J., and Mack, R.L. (Eds.) Usability Inspection Methods. John Wiley & Sons, New York, NY, 1994.

[4] Van Galen, G.P. and Müller, M. (Dutch, English summary) Repetitive Strain Injury, Stress, Muscle Tension and the Neuromotor Noise Concept. Tijdschrift voor Ergonomie, nr.2 (1995) pp. 3-17, NVvE.