How To Live With Software Problems

Klaus Jul Jeppesen
The IT University of Copenhagen, Glentevej 67, DK-2400 Copenhagen NV, Denmark
+45 38 16 88 89
klausjj@it-c.dk

Abstract: In general, software systems are relatively error free and support us doing our work. We could not live without these systems if we want to maintain the level of service, to which our customers have become accustomed. Most users, however, from time to time experience problems. It seems that despite all improvement efforts in areas such as specification, usability, testing and so on, software in use will cause problems, and we must find ways to live with this fact.

From a user perspective a system is either in the phase of implementation or operation. This paper focuses on the problems encountered by users when a system is in the operational phase and the development organisation is not available for problem resolution. The goal must be to support the users to get the highest possible benefit from the system. We must help users find the best way to live with the system.

This paper describes how to record problems and the basic principles to follow when these problems are processed. Examples are used to show how the potential benefit can be estimated. This forms the base for a decision on whether it is worth to cure the problem or not. It is common sense to provide support in this way, but experience shows that many organisations do it in an unstructured way and without recording the costs and benefits. Many provide support only because the cost of task failure, due to system problems, is high.

The paper then presents various measures to be taken to help the users do their work efficiently and get the best out of the system. It will be demonstrated how many of the problems can be cured through better information to the users, better work procedures, and system tailoring - without modifying source code.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35404-0_19
M. A. Ardis et al. (eds.), *Diffusing Software Product and Process Innovations*

1. BACKGROUND

Computer systems have the potential to substantially improve work practices and allow users to develop organisation to which they belong. As Pressman [5] describes, software is a key technology and software-based systems are used in many places. Most companies could not operate without computer systems and maintain the level of service expected or assumed by customers.

The author has been working for 20 years within the field of development as well as that of operation of an installation with several hundred users. It is my experience as a manager of software development that the development process, by now, is fairly well described, and when the suggested practices are followed, the result can be a high-quality system. On the other hand, during my time as a manager of IT-support, I have frequently observed users encountering problems. This has caused them a lot of frustration, because small features of the system have given them a lot of inconvenience or annoyance. I noticed that often a small action by someone, who happened to be there and knew what to do, was most helpful to the user. Gradually I developed the idea that we must accept the fact that users encounter problems, and that we can improve the users' situation, if we address the problems adequately.

The 70s and 80s brought many stories of errors in software systems. Some labelled it a "software crisis" (e.g. Pressman [5]). Many studies and much effort have been made to deliver software error free. Techniques for establishing specifications have been improved. Usability methods are used to ensure that systems will satisfy the users. Designs are reviewed, and modern programming languages are supposed to ensure a correct implementation. Rigorous testing verifies that the system operates as specified. Not that software, in general, is totally free of errors, but the situation has improved. As Wirth [6] stated: "There is an enormous amount of software in this world, here and everywhere, that works perfectly well and does its job".

In spite of improved software engineering, resulting in higher quality software, the users complain because they encounter problems during operation. They often experience situations where the system makes work awkward or in some cases makes it difficult for them to complete their task successfully. These problems range from mild annoyance to outright faults causing task failures. It is time to study the operational phase in more detail and find better ways to support it.

Traditionally, usability is discussed in the context of software specification and development. The usability, however, can only be finally evaluated, when the system is in the operational phase. Then it may be too

late to change the software and we must seek other solutions to reduce the number and the impact of user-experienced problems. Most users can tell at least one story about a problem in a system they use daily. Practitioners in the field of systems operation confirm that users still experience problems. The software seldom malfunctions, as this kind of errors typically has been removed during development. However something annoys or bothers the user. An example is a user who has forgotten a password or forgotten to change it before it expired. The user suddenly cannot gain access to the system at all. Another example is a user, who consecutively enters records via a screen, and the screen is cleared between each record, so several record fields have to be re-entered although they contain the same data as the previous record. A further example is the user who computes a report manually, not knowing that the system can do it. These examples are typical, and together with a few more, they will be analysed in the following sections.

Often the system is based on commercial off the shelf (COTS) software. In this case, we have little or no influence on the details of the software specification. Furthermore McKinney [2] reports that a drawback of COTS software is that the availability of features and bug fixes is related to releases, and that a new version "often swells resource requirements significantly", which may prohibit the use of a new release. The functionality of COTS systems may not fit perfectly, but we use it anyway and have to use it as is.

A rich literature describes the software development and the various aspects of the software engineering. The software life cycle process standard defines all the activities including resolution of problems. But the 'user-support' process is only briefly mentioned as part of the IEEE standard [4] and the general literature on the subject is limited. The standard is limited to error reports that are fed back into the development process. In this paper we focus on the case where this is not possible. During the operation phase there is often no communication path back to development. In general the operations phase deserves a detailed discussion.

Through better support, many of the user's problems can be avoided or a solution provided, so that at least the user can live with the problem and achieve an acceptable level of productivity. Due to lack of knowledge, the users typically do not utilise the full benefit of the system capabilities. Again, better support could help.

Beizer [1] categorises errors according to the source. This relates the problem to activities in the development process and is valuable, when discussing how to improve development. Here, however, we focus on actions to be taken after the problem has been identified and after development is supposed to be complete and the system is in operation.

During this phase we are interested in the type of action that can solve the problem.

This paper suggests that the problem processing should comprise the following steps: First recognise the problem and make a proper recording, then make a rough estimate of the potential benefits if the problem is removed. Such an estimate may show that the benefit is small and that there is no reason for further action. It may on the other hand reveal an interesting potential benefit. If so, possible actions with associated costs must be explored and an optimal set of actions found.

2. PROBLEMS ENCOUNTERED BY USERS

The following examples illustrate the problem types that users encounter. We will later see how many of the problems can be resolved by the support organisation. The common aspect of the examples is that the user experiences a problem. The root of the problem can be either the user, the system, or indeterminable which is, however, not that significant. The point is that users feel they need assistance or system changes to get their work done efficiently.

Each problem has its own set of solutions. Some are temporary workarounds and others solve the problem. Optimal solutions may not be available or the cost to get them may be too high. But the user should be left with the feeling, that the best possible solution has been achieved.

2.1 Examples of Problems

The examples describe cases experienced by the author. Many more examples could be given, but these few are sufficient to demonstrate the scope of problems encountered by users. The benefit of various solutions is presented in the following section.

1. **Password**. The users forget their password or neglect changing it when required (often despite warnings given by the system). When the user arrives in the morning, the system is blocked, so the user cannot log on to the PC, and no work at all can be accomplished.
2. **Office change**. Another example is an organisation where the employees frequently move physically between offices. When they move, the IT-department has to change some of the set-up of the PC before the user can get access to the computer network. The average response time from the IT-department is 6 hours, and support cannot be initiated before the move. The user can not use the PC the first day in the new office.

3. **Mismatching part numbers**. A company uses one system for bookkeeping and another for production planning and storage management. Once every day the two systems exchange information to synchronise sales, production, purchase of parts, and shipping of items produced. One day the bookkeeping system was upgraded to a new version with a different part numbering data structure – and the two systems could not exchange data, because the part numbers did not match. The production came to a halt after some time.
4. **Missing report**. The user is a project leader in an engineering department. When controlling costs of an order, the project leader must enter the order and print out a page for each cost item on the order and then manually calculate the total and the difference from the forecast. The project leader spends quite some time, and occasionally makes a mistake in the manual calculation.
5. **Entering payment information**. The user is entering payments in an accounting system. For each payment the screen is cleared, and the user must re-enter all information, including the date, the user identification etc. This is additional work that the user feels is a waste of time.

The potential actions against the described problems are listed in section 6.

2.2 Benefit

The benefit of curing a problem depends on the organisational structure and the type of work. To get some idea of the approach, we use an organisation I have worked in. The users were:

- 10 project leaders
- 50 staff employed in general administration (bookkeeping)
- 60 administrative assistants
- 200 technical computer users
- 10 users in supply management etc.
- 40 staff employed in production of goods, storage handling, and shipping
- 30 sales persons.

This gives a total of 400 PC users. The examples below demonstrate how carefully you must calculate the benefit before deciding a solution, as the conclusions in some cases are surprising.

1. **Password**. Let us assume that the password must be changed once a month. 5% of the users (20 persons a month) forget to do this. They spend 10 minutes getting support staff to enable them to log on. The organisation could benefit 10 minutes times 20 per month, which is 10 minutes per working day. From a pure economic evaluation this is not worth spending any effort on. But if you are sensitive to employee

satisfaction, something should be done. You might simply mention the rationale of password changes to the users in the next IT information letter and encourage them to do as requested.

2. **Office change**. Let us assume that the project groups are moving every 6 months on average and that half of the technical staff is involved. If each person waste one day per office change 100 days are lost every half-year – equivalent to one full-time person. It would obviously be a good investment to make a guide so that the technical staff can make the set-up themselves.
3. **Mismatching part numbers**. When a system supporting the company infrastructure fails, the damage may be substantial. The lost working hours is only part of the cost. The loss of business and the disruption of production may be fatal for the company. The upgrade of the book-keeping system must be undone, or the production system must be upgraded so that the part numbers can be synchronised. We are facing here an example of a problem to which a solution must be found immediately, and which support staff ought to prevent from occurring at all.
4. **Missing report**. Assume that the project leaders must report monthly, that each leader has 10 projects, and that the manual calculation takes one hour. This corresponds to 100 hours per month. In the actual case, it turned out that a report used by bookkeeping provides the information, so the function can be provided without any change to the system. There is substantial benefit, 100 hours per month and reduced employee annoyance.
5. **Entering payment information**. Assume the organisation makes 60,000 purchases per year. The saving by tailoring the system is perhaps only 30 seconds per transaction. However this totals approximately 40 hours per month. Again, a small change can bring a substantial benefit.

3. SOLUTION SPACE

There is no universal key to problem resolution. The actions described in the following sections must be seen as a set of ideas based on lessons learned from solving problems as described in the examples. A support organisation should make its own list of actions tailored to the applications currently in use. The list should be enhanced while the support staff gains experience in connection with the type of problems typically encountered. The possible actions are described in four groups:

- The first group of actions addresses the problems that are basically caused by the user's lack of knowledge due to ignorance, poor

documentation, lack of training etc. Short-term resolution is normally possible by advising the user. In the long-term the general knowledge level must be increased or the system must be made easier to use in order to avoid other users experiencing the same problem.

- The second group solves problems through a change of procedures and rules.
- The third group solves problems through change of system parameters. Many applications are built on standard software tailored to the individual user or a group of users, which makes it possible to change the functionality without changing the program code. An important aspect of this is that it can often be done on the fly or with only minor interruption.
- Finally the support staff cannot resolve some problems. Resolution must be referred to someone with better knowledge and access to more specialised tools, or resolution may require special skills, e.g. programming. Typically, this type of action is more costly, and the cost/benefit of problem resolution becomes an important issue.

3.1 Inform Users

Many problems arise simply because the user does not know how to use the system or why the system requires operations to be performed in a certain way. The erroneous use of the system leads to wrong results or failures because the system is used in a way never intended or tested.

The supplied standard documentation is often related to the product and not to the specific context, in which it is used. The documentation often resembles a guide to a toolbox. The support staff may find such documents useful, but that type of information is not sufficient to support the user's actual work practice. The support organisation must ensure the availability of user-relevant documentation, which can be in the form of short instructions or recommendations. For instance a checklist of tasks, which the user routinely must perform, could be provided

The support staff must foster an environment, where users will admit problems and ask questions. The support organisation could establish and maintain a database with frequently asked questions. Support staff would be responsible for the maintenance, e.g. providing correct and accurate answers and removing irrelevant questions. Users should be allowed to enter information into the database, which will allow users to learn from each other. A user might have a problem to which support staff does not have a solution, but some other user does. It is the responsibility of support staff to have this information reviewed and to prevent bad habits from spreading. Another database could give information about known problems and the

intended resolution. The cost of maintaining such information systems must, however, be carefully considered.

Yet another approach could be to establish "ambassadors" of the system easily accessible to the users. They are often called "super-users", as they are actually using the system. They ensure a close contact with users and immediate responses to problems. Super-users work in the user environment and are consequently close to the users and their problems and frustrations. One must, however, ensure that the super-users are well trained and kept up-to-date on problems and their solution. The recording becomes more important, but difficult to enforce, as the support staff is distributed in the organisation.

3.2 Operational Procedures

Support staff can assist the user in organising work practice. Often some functionality is available through several different operational procedures. By interviewing or observing some users, the optimal operation may be determined and then communicated as best practice to all users. Storyboards may be used to discuss better work practices with users. A forum of users could be created, where procedures and rules can be discussed and elaborated in order to find the best way to live with the system or to utilise a certain function. Such a forum can also be used to test ideas for new procedures. For some problems support staff can provide a workaround. This will typically consist in a change in the operation of a function or in a recommendation to use other features of the system to achieve the desired result.

3.3 Tailoring

Tailoring denotes the setting of parameters to make a standard product or a framework system function in a certain way. In some systems the set of parameters is huge, and the setting non-trivial, because some parameters are interrelated. Support staff should be capable of performing some tailoring and in the non-trivial cases possibly communicate with somebody with more competence. To control the tailoring, support staff is responsible for configuration management, which ensures that modifications are properly documented and tested, before they are put into operation.

Likewise support staff must be responsible for configuration management of the software components. Support staff must participate in the testing of new versions or modifications to the system. This has got three purposes: First, it can prevent cases, where the system fails totally. Second, performing the system test is a good training as regards the system functionality and

operation. Finally, support staff will have more confidence in the system, when they have witnessed the test and know that the system works, at least with the test set-up.

It should be noted that much of this tailoring and configuration management equals software development. The good practices employed during the development processes should therefore be applied during support as well. Tailoring must be properly specified, documented, tested, and the changes and test results carefully recorded.

3.4 Escalate resolution

When the user problem involves a program error or missing functionality, the tailoring may sometimes provide a solution. In other cases, however, it requires a modification of the source code. Programming and tailoring is often outside the capability of support staff. In these cases support staff have to pass the problem to a third party, with the required skills.

Some problems are difficult to resolve, and the process may require time and money. In such cases the actions often have to be decided by management after a more thorough evaluation of the benefits compared to the costs. It should be the responsibility of support staff that all information required to make this decision is obtained from appropriately qualified sources.

In many cases, however, support staff **must** provide a solution, at least in the form of an acceptable work-around. This could be the case, when a problem, which is critical for operation, has occurred, and an immediate solution is required, or when escalation is not possible. Some examples are:

- there is no access to the software development,
- the original supplier does not have a support function,
- the time constraints do not permit the optimal solution to be found.

The only possibility for support staff is then to establish a usable solution by combining actions from the previously mentioned groups, i.e. through user information, change of procedures, or tailoring.

4. STRUCTURING THE SUPPORT PROCESS

Two aspects of the support process are discussed in the following. The first concerns the activities of the resolution process and the second the registration of all reported problems and the actions taken. More work is required to make a more complete description of the support process.

4.1 Resolution process

The goal of the support organisation is to have problems resolved quickly and efficiently. A quick solution that creates a new problem may, however, be worse than no solution.

The problem resolution process consists of steps similar to the processing of anomalies, described in the IEEE standard [3]. The steps are as follows:

a) Detect and report the problem,
b) Analyse the problem,
c) Develop a solution,
d) Deliver the solution to the user.

The support process starts, when the user experiences a problem or somebody observes the user practice and detects that user performance could be improved. It is important that the user recognises the cases where something is a problem, and that the user prepares a proper report. The environment should help the user make reliable, accurate, descriptive, and consistent problem reports.

Users must see a benefit from reporting problems; otherwise they will feel it is a waste of time. To obtain such an approach requires the right attitude, the right tools, and the right training of support staff. The support process must be visible, and the results well communicated to the users.

The support staff must consist of well-educated and trained personnel. Support staff must know the systems at least as well as, and preferably better than, the users. They must be motivated to provide competent responses to user complaints.

Support staff must have appropriate tools to process the problems. The tools are used to analyse the problem and assist in providing solutions. This could, for instance, be a tool with remote access to the user's PC, enabling support staff to see what the user is doing and provide guidance to the user via telephone. It could be a tool to track all transactions and detect bottlenecks, or it could be a separate "development system" to experiment with different tailoring or user procedures. The required tools will depend on the application, and support staff must participate in determining, what is appropriate.

Quality assurance must be applied to the support process. It has often been experienced that the user is asking for a quick solution, and that support staff comes up with a quick fix. This is tempting and rewarding when successful, and terrible when it creates new problems. To be successful in the long run, support staff must ensure a proper quality. Techniques from software development in terms of reviews and testing should be applied to ensure success.

When the problem has been resolved, the solution must be rolled out to the user. This must be done in a professional way, and the solution presented to the user without making the user feel incompetent. The solution must also be distributed to other users, so that they know what to do, if they encounter the same problem, or how to prevent it from occurring at all.

4.2 Registration and Analysis

All support requests should be recorded. The registration system or tool must be made very easy to use, so that it does not constitute a barrier to reporting. Registration must for each problem record a short description of the problem and later the resolution. Furthermore, the time of the event, the time of the resolution, and the time spent should be recorded.

The data collection serves several purposes. The registration will provide data for the evaluation of benefits, for the prevention of problems, for the quality assurance, and for the strategy for long-term evolution of the system. It may also be used to assess the quality of the support organisation.

A prime purpose of the registration is to drive the prevention of problems. Analysing the different types of problems and the frequency or repetition of problems can be used to identify the need for better user education, and for implementation of modifications to the system. The recording should be organised so that it is easy and fast for support staff to find out whether a new report describes a new problem or a problem already recorded, and potentially solved. Support staff could also provide a 'first-aid', which should describe the most frequently asked questions, and the answer to these. Support staff should propose additional user training, since support staff knows where the problems are.

The records should be used for an assessment of the application used. Applications with frequent problem reports are candidates for a closer inspection. The impact of the problems on the work should be evaluated, and the cause of the large number of problems should be analysed. The knowledge obtained can be used to make decisions on the short-term development and enhancement, and to make strategic decisions concerning the long-term evolution of the system, or a potential replacement.

Finally, the records can be used to evaluate the quality of the support function itself. These records can be used to examine whether responses cause new problems, were wrong, or did not provide optimal solutions. The records can also form the basis for statistics on the average response times, the lead times, the total resolution time etc. Special care must be exercised when these figures are being analysed, as they can easily reveal information about individuals. For instance missing efficiency by support staff, or disclosure of individual users, who report many problems caused by their

own bad performance. The statistics must be made public in a way that hides all personal details.

5. THE BENEFIT

As was illustrated above, user support can be a substantial benefit to the organisation. The benefit will depend on the application as well as on the organisation, and also on the knowledge level among the users and support staff. Furthermore, user support may give indirect benefits, which are difficult to calculate. Benefit will vary during the lifetime of the application. When users start using the system, they will have many questions, typically simple to answer. Later, when they have gained experience, and some tailoring has been done, problem frequency will decrease. If support is not reduced correspondingly, costs may outweigh benefits.

The most significant indirect benefit is increased user satisfaction. It is commonly agreed, that a high morale and good general well-being at work increases productivity. For some applications it also leads to better customer service, which provides several advantages to the organisation. The indirect benefits must be estimated individually on a case by case basis.

The first consequence of letting support staff solve the problems is that effort and cost are moved from the user group to the support group. We must, however, assume that the support staff, if better trained and having better tools, can solve the problems more efficiently than the users. Hence support staff should spend less time on resolving problems than the user would have to do, and the overall effort used on problems should be reduced.

The real benefit comes from the activities mentioned above. Establishment of better procedures, tailoring etc. are one-time efforts to be made by support staff, but this will potentially benefit all users repeatedly. Not only does the number of users multiply this saving. The saving is recurring every day over the lifetime of the system. This is clearly illustrated in the benefit calculation for the missing report and entering of payments.

The users benefit the most, when they conceive the system as a good and pleasant system, assisting them in accomplishing their work in a productive manner. So investment in improved user knowledge also benefits. To ensure a high level of knowledge, it should be made attractive to the users to spend some time on training, studying manuals, and understanding recommendations.

6. ANALYSIS OF THE EXAMPLES

This section compares the four types of actions against the five sample problems in section 2.1. In table 1 each column represents one type of action. A given problem resolution can be achieved by actions from one or more columns. Each row represents one of the examples, and the cells give a short description of possible actions. The cost of actions differs and the cost/benefit of the problem resolution must be considered when appropriate action(s) are to be chosen.

As can be seen from the table 1, a given problem may be addressed in several ways. The problem with entry of payment information, for instance, has two potential actions. If some users work best when the screen is cleared, the individual users should be trained to find the possibility of making the appropriate selection. However if no users profit from this selection it is better to change the functionality system-wide through tailoring. Support staff must select the appropriate action(s). One action may provide a short-term solution, and another action may be required for a better long-term solution.

7. CONCLUSION

Personal experience and discussions with other practitioners support the assumption that no matter how well we develop software, there will be problems to deal with during system operation. Users encounter problems ranging from small annoyances to functions that are cumbersome to utilise.

First of all, it is important that all problems are carefully registered. The recorded information can be used to estimate the benefit of solving the problem and used for a general assessment of the system in use.

The idea is that the elimination of time-wasting problems is the key to success. The examples in the paper demonstrate that in many cases the elimination of problems result in a substantial benefit to the users. The greatest benefit is achieved, when support staff plays an active role in preventing the problems in the first place.

Table 1. Possible actions for the sample problems

Action type / Example	Information	Procedure	Tailoring	Escalate
Password	Make a task list for the users to remind them to check for password change.	Consider change of password period.		
New office	Enable the user to make the changes themselves.	Write a procedure for the changes to be made, when staff changes office.		Consider automating the set-up of computers on the network.
Mismatch part numbers		Ensure proper change and test procedures.	Make configuration management, test before operate, and apply development tools.	
Missing calculation	Ensure exchange of best practices between departments.	Establish procedure for project reporting.	Use tailoring to provide the required reports – if possible.	Get external assistance to provide the required reports.
Entering payment information	Inform users on the possibility of selection of functionality.		Change functionality through parameter setting.	

When a benefit has been identified, the next step is to find the appropriate actions. They must provide a useful solution, and the costs must be justifiable by the benefit. The paper describes four types of actions:

- Inform users. In some cases, benefit will be achieved by providing better information to the users and to ensure that users exchange information. The information should help users avoid problems and find the best way to use the system. Guides, written specifically for the actual application, are more relevant to the users than the standard guides typically provided for software products.
- Work procedures. In other cases work practice can be investigated and developed in co-operation with the users. Users should be guided to use the system in a way that best supports their work.
- Tailor the system. The system can be tailored to the users actual needs. It must be accepted that tailoring is a kind of programming, and that good practices from software development, such as configuration management and testing, must be applied.
- Escalate resolution. Support staff cannot solve some problems and the resolution must be escalated. Often a work-around must be provided and the users will benefit even from a non-optimal solution instead of none at all.

The study shows that data must be collected to gain a better understanding of the problem types experienced by users. Such data would allow us to better estimate the potential benefits and suggest a more systematic selection of problem-solving actions. The big challenge may then be to get practitioners to use these recommendations.

REFERENCES

[1] Boris Beizer: Software Testing Techniques, Second Edition, Van Nostrand Reinhold, New York, 1990

[2] Dorothy McKinney: Impact of Commercial Off-The-Shelf (COTS) Software on the Interface between Systems and Software Engineering, proceedings of the 21st International Conference On Software Engineering, May 1999, Los Angeles, California, USA.

[3] IEEE Std. 1044-1993, IEEE Standard Classification for Software Anomalies, December 1993.

[4] IEEE Std. 1074-1997, IEEE Standard for Developing Software Life Cycle Processes, December 1997.

[5] Roger S. Pressman: Software Engineering – A Practitioner's Approach, Fifth Edition, McGraw-Hill, Berkshire, 2000.

[6] Niklaus Wirth: Software – Quality or Quantity, That is the Question, Proceedings of conference on Managing Software Quality Engineering Success, January 1997, Baden, Switzerland.