Role of Human Emotions in Requirements Management

Sreevalli Radhika. T.
Robert Bosch India Ltd.

Requirements management is an area where satisfactory results are not seen even after establishing well-defined processes and adopting good tools. This article looks at this problem from a totally different angle and finds facts that are normally not observed. It discusses the effect of human emotions in requirements management.

Many people stereotype software developers as emotionless individuals who have more in common with their computers than with their fellow workers. One thing brings out emotions in software developers more than anything else: requirements. Organizations try to establish defined processes for activities related to software development, and to achieve results that are fairly independent of various parameters (i.e., people, location, time) involved in accomplishing the desired activity. Generally, a reasonable success is assured from most of the processes with a few exceptions. One such exception is *requirements management*. This is because of the role emotions play in the acceptance of requirements. The effect of the emotional response to requirements can make a major impact on how software is developed.

Even after establishing a well-defined process and adopting a tool for managing requirements, it is often found that managing requirements is not enough. The main reason for this is understood with three human factors that generally affect the quality of any work:

- The way people act (work).
- The way people think.
- The way people feel.

The first two factors are normally noticed by project teams and are addressed by identifying required *tools* and *processes*. Tools and processes drive the actions and thinking process involved in the different phases of software development. They do not take into account the way people feel. Typically, there is no special attention given to this third factor. It is not apparent how many activities are directly affected by the emotional response of the developers. Most of the time, it is possible to achieve good results by taking care of the first two factors. However, even good results can be affected by emotions.

Other processes are not as affected by emotions; for example, consider configuration management. Configuration management usually does not provide opportunities where different people can *feel* differently about the handling of configuration items. There is not much room for different opinions to be formed in the way configuration items are identified and controlled. In general, configuration management requires specific actions rather than detailed consideration and analysis. As a result, it is not subject to the effects of emotions in the way that requirements management can be. Only the first factor mentioned above – the way in which people act – is important here and the differences in this factor can be avoided by defining a clear process for configuration management. Once a process is defined, the project team can just follow those steps. The *thinking* and *feeling* factors will come into play when considering how to improve the configuration management process.

The same is not true with requirements management. For this process, the thinking and feeling cannot be avoided. A well-defined process can only address so much, because there is something that affects the thoughts and feelings of the developers – *the requirements document*. The very existence of this particular document will have a continuous effect on the thoughts and feelings of the team members. The effect of this document continues through the life cycle regardless whether the document changes frequently, whether it is baselined, or whether it is maintained under a proper version control mechanism.

It is okay that the requirements document elicits emotions. A requirements document should have the power to enable the thoughts and feelings of the team members. What is important is the direction this document takes the reactions of the team members. It should help to bring innovation and motivation among the team members, rather than bringing irritation and frustration.

Studies about requirements documents show that the requirements usually have problems, including omissions, contradictions, ambiguities, duplications, inaccuracies, too much design, and irrelevant information. This may be true, but as far as the project team is concerned the emotions related to these issues are more serious than the fact that the problems exist. While processes may be in place to handle the requirements, they do not matter as much as the reactions of the developers.

It is easy to establish the physical traceability from requirements to all the affected documents in the project. Tools help in that task. However, the traceability from the author's intent to the final product does not occur as easily. Tools cannot help here. It is up to the relationship between the author and the reader. That relationship is a product of the respect the two have for each other and for the requirements document.

Lack of Respect for the Document

Almost everyone accepts the importance of communication and the effect it has on the human emotions and relations. If a day starts with a good incident, its effect remains the entire day. The same is true in case of the opposite situation. If the first incident of the day is bad, that effect lingers the rest of the day.

The same rule applies to the requirements document. When you consider the communication involved in a project, everything starts with the requirements document; the entire project depends on this document.

It is not only essential for this document to be clear and correct, but it should also gain the respect of the project team. If a requirement is clearly defined at this point, it will be accepted and respected by the software developers. If it is not, problems can ensue. Even something like a small contradiction between different sections of the document can deteriorate the respect for the document. This can be true even if it is a small contradiction that may not damage the clarity of the requirement as a whole.

If the requirements are not clear, the reader can start assuming things wherever a little bit of ambiguity exists. People tend to fill in the blanks. Often, they do not

even realize they are assuming things that are not there. In such situations, software developers will think they are capable of understanding a poorly written document. This reaction can cause problems down the road as assumptions can diverge from the original intent of the document.

A number of things can cause a lack of respect for the document. These include spelling mistakes, improper organization of the contents, and redundant statements. Authors may ignore small mistakes like the wrong date or the wrong version number. The readers will not. All of these items add up. If the readers disrespect the document, it can lead to frustration and anger. It will affect the actions they take in developing the requirements.

Some of these points may not seem important while preparing the document, but they can actually pass an indirect message to the reader about the document. The author should always consider the reaction of the reader. If the author takes care to avoid these little problems, it can bring about later benefits.

Lack of Respect for the Reader

A lack of respect for the reader ties closely to lack of respect for the document. If the author has a high respect for the reader, it will be apparent in every small part of the document. If the reader feels the respect of the author, he or she is more likely to accept the requirements and work with the author on future considerations.

An author can show his respect toward the reader in many ways. These include giving appropriate information at appropriate places and not leaving any loose ends. A reader who feels the document is giving him critical information – but not strict instructions – will feel more freedom in developing the requirements. This leaves the reader feeling he or she has an important contribution to make and is likely to gain project buy-in. Basically, the care the author takes in developing the document can make the reader feel better about the requirements document.

On the other hand, even a little carelessness in preparing the document can have a very negative effect. Things like improper formatting or inconsistency in font size can be as irritating as the errors mentioned above. If the reader feels the author was careless about these little things, he or she can feel a lack of respect from the author. This can create a reciprocal lack of respect for the document.

When the document is prepared with utmost care, it can demand respect from the reader. The reader will also feel respected. The reader will take more care in analyzing the document and will be more likely to work with the author to ensure everything is correct. That mutual respect can be a strong bond that will continue through the life of the project. Then the reader will more likely try to understand the document in detail and be more likely to cooperate with the author. Otherwise he or she will always see something wrong in the document.

When the reader does not have respect for the document or the way in which the document is written, then the reader's own point of view comes into play. It can often be very different from the requirements specified in the document. This can lead to a dislike or disregard for the author's point of view.

"Tools and processes drive the actions and thinking process involved in the different phases of software development. They do not take into account the way people feel."

Whoever the reader may be, once convinced of the quality of the requirement or lack thereof, the quality of the subsequent work will be affected by the reader's response to the requirement.

Lack of Respect for the Author

When the reader does not have enough confidence in the author who has prepared the specifications, the requirements will not get the consideration they need. Normally the author's background is not shown in the requirements document. However, the reader's past history with the author can color his or her reaction. A bad history increases the likelihood that he or she will expect difficulties and view the quality of the document with skepticism.

If readers have more technical knowledge than the author does, they may not read the document with the same point of view as the author. Such readers may quickly conclude that the document is not correct. Instead of finding the reason behind that, readers can assume that the requirement is wrong due to the author's ignorance and lack of skill.

Once the developers conclude that the author is not to be respected, the readers' analytical and technical capabilities will go in a direction of proving that the requirements are not feasible. Subsequent requirements from the same author are likely to be dismissed in the same way.

It is not that they intentionally consider the requirements this way, but it is difficult to overcome that bad initial reaction. Unless they see the reason, logic, and intention behind the requirement, they will never be able to succeed in implementing it as desired. If they are predisposed to disrespect the author, they are not likely to work with the author to ensure the requirements are implemented the right way.

All this does not mean that people should not give their comments on the document. The point here is that it should be done in a reasonable way. Putting some structure to the requirements review and analysis procedures will help with this, but will not solve all the problems caused by the lack of respect for the author. To understand the requirements, it is necessary to believe that there is some reason for them to be written as they are. To believe that such a reason exists, it is necessary to have respect for their creator. Cross training between the requirements' authors and the developers can help them understand each other's abilities and constraints. This can lead to more communication and understanding, which can only lead to better development and better results.

Two more things can bring out the emotions of the people involved: changes to the requirements, and the way in which change requests are handled.

Changes to the Requirements

Many times the freezing of requirements does not happen in time because of changes to the requirements after the project team has begun work. Scope creep happens, but it can lead to major frustration and even resentment on the part of the project team. If a lot of changes happen after the project team has started reading and reviewing the requirements, the team can lose faith in the project. This is especially true if changes to the requirements document take place after the design is started.

Most project teams see some risks and problems due to shifting requirements. If the requirements are not frozen and are changed many times, there may be lot of rework that, in turn, results in added effort and schedule variances. Unexpected

changes bring frustration and can lead to conflict between the requirements' author and the development team. It adds difficulty as the team tries to maintain various versions of the documents properly. Juggling the constant changes may lead to poor quality

However, these problems can be managed to some extent with the help of good processes and tools. But there are some aspects related to human emotions that should be considered more risky and difficult to handle.

As mentioned earlier, the project team should develop a liking or at least respectful acceptance toward the requirements. This will help the team achieve better design and a better quality work product. This is not possible unless the requirements are frozen. As long as there is a possibility to change the requirements document, there will be a feeling that the requirements can still be improved. It may also lead the project team to feel that the requirements will never be right.

Most developers have been on a project where there was a major delay in freezing the document. Let us use an example where development had to start when only 60 percent of the requirements were clear, and the requirements document was still undergoing changes. In such an instance, there can be a lot of suggestions for how to address the requirements. If the requirements were vague enough, they can bring out a number of suggestions. The team will then have to work through each of these alternatives to determine what was really desired by the requirements' author. Give and take with the author at this point can produce a series of requirement changes. It can be difficult to manage the suggestions raised within the project team when this occurs. As a result of their ability to make changes to the requirements, the developers will keep adding their desires to the requirements.

The problem here is that instead of spending the efforts on improving the designs, the project team spends its efforts on improving the requirements. No one assigned this task to them, but the team's frustration with the vague requirements will make them take it on themselves. This creates more work for them and for the requirements' author. The team spends more time on the requirements' developer's task than on their own assignments for no advantage. The rework adds time and cost to the project when it could have been avoided early on.

To alleviate this problem, the team should work with the requirements' developer to reach an understanding early on in the project. Requirements reviews can reduce the project team's frustration and help build a relationship between the author and the developers that can influence future projects. Once a solution is found, the requirements should be frozen and the changes limited to fixing problems.

The Way in Which Change Requests Are Handled

Regardless of the delay in freezing the requirements document, there will always be the possibility of getting change requests during the life cycle of the project. There are some difficulties that are specific to change request handling. These include capturing the change requests in a systematic way, evaluating the impact of these changes on the current development, and tracking them properly throughout the development life cycle. Mistakes in any of these tasks will be harmful to the project and are typically taken care of in the processes.

"A large number of changes bring out the frustration of developers because they feel they are trying to hit a moving target."

But again, the processes can only address the problems related to the first two human factors discussed in the beginning of this article. Mistakes in these kinds of tasks can happen if either the *action* or *thinking* of the team is not in a systematic or organized manner. These mistakes can also happen because of the feelings of the people involved. If the developers are already frustrated or irritated with the requirements document, their feelings can drive their behavior when handling changes to the requirements. If developers are found to be deviating from a process, it is necessary to consider not only the risks due to process considerations, but also the problems that may arise due to feelings.

If there is no process defined for handling the change requests or if the defined process is not followed consistently, then it will lead to various assumptions and negative feelings among the team members. People may take it as favoritism or injustice when the criteria for accepting or rejecting a change request are not apparent. If the behavior of the author or developers is not clearly understood, hard feelings may result. If the procedures and guidelines are not defined and practiced consistently, then the reason behind approving and rejecting the change requests will not be understood correctly by the team.

A large number of changes bring out the frustration of developers because they feel they are trying to hit a moving target. By the same token, if the changes are not handled properly, they can increase the level of frustration.

Conclusion

Human emotions play a very important role in requirements management. Developers react to requirements in a number of ways. The range of emotions they generate come from a variety of reasons. Poor requirements can elicit frustration, irritation, anger, and disrespect. Good requirements can bring acceptance, understanding, and buy-in. Processes can be defined for capturing, analyzing, reviewing, implementing, and verifying the requirements. Tools can be identified for tracking and implementing the requirements without errors. However, if human emotions and feelings are not addressed, then the desired result may not be achieved in spite of the use of those processes and tools.

Frustration and confusion over requirements can lead to unexpected behavior by development staff. When someone behaves in an unexpected way, people will question their behavior. There may be an attempt to change that behavior. But there should definitely be another consideration, which is *understanding the reason behind the current behavior*. Often, that reason is tied to the emotional response of the person.

How can an organization deal with emotional reactions to requirements problems? The best way is to take the emotions out of the process as much as possible. Many of these problems do not need a separate solution. Identification of the problem itself can lead to a quick solution in many cases. However, the following considerations can be adopted to improve the existing and established processes in this respect:

- A consistent and well-defined requirements definition process can help by taking the focus off the people and onto the process. This starts with a template for the requirements docu-

ment that covers information about the author and the reasons behind a particular requirement.

- Maximum care taken while preparing the requirements document will help avoid overlooking even the very small points like spelling mistakes that may deteriorate the respect on the document. Peer reviews can help catch these kinds of errors and ensure that the requirements document meets standards.
- The change request handling process must be clear, quick, and consistent. It should not be very easy to add a change request to the document. There should definitely be a three-step process like initiation, evaluation, and approval. These steps should be carried out quickly, but they should not be skipped.
- Involvement of the development team in reviewing the requirements early in the process will establish communications between the author and the developers. Open communication between the groups will make them all feel they are part of a team, and they are more likely to try to reach a mutually satisfactory result.

To achieve best results, the project team members should feel a part of the requirements process and should develop an involvement with the requirements. If they have a true stake in the results, their emotions will be guided toward achieving a common goal. Defined processes and support tools should complement their efforts and improve their possibilities for success.◆

About the Author

Sreevalli Radhika. T. works for Robert Bosch India Ltd. as a department quality coordinator. She has been involved in the management of embedded projects for the past decade. Radhika is also interested in fiction writing and has published more than 50 articles. She has a post-graduate degree in electronics and communication engineering.

Robert Bosch India Ltd.
Phone: (91-80) 2299-9052
Fax: (91-80) 2299-9156
E-mail: radhika.sreevalli@in.bosch.com
tsradhika@rediffmail.com

Letter to the Editor

Dear CrossTalk Editor,

Regarding the From the Sponsor article by Tom Christian in the August 2005 CrossTalk. The first paragraph of his article ends:

> "... relentless commitment to quality: employing peer reviews, configuration control, documentation, and testing."

Although these items are all necessary to achieve quality deliverable software, they are not sufficient: The one key item missing from the above list is, in my view, the most important one, good design practices.

I have heard said numerous times over the years, "You cannot test in quality," and it is so true. A team can spin its wheels for months thoroughly testing a system only to find itself retesting, retesting, retesting because every change seems to *break* the system in unintended ways. This is usually because the basic design of the system is flawed due to one or more of the following practices: use of global variables, lack of cohesion, close coupling, inadequate abstraction, lack of encapsulation techniques, etc. (All these principles I mention pre-date object orientation, yet it is surprising how little they are understood even today!)

A system with a truly good design could possibly succeed with limited testing, documentation, and peer reviews (configuration management is always crucial in my view). But, a poorly designed system will fail no matter how much it is tested, reviewed, or documented.

The larger and more complex the system, the more crucial it is to use sound design practices. It does not come automatically. No specific software language can guarantee it. It is a much larger challenge than "properly indenting your code." It is sorely needed today more than ever.

Robert Wolfman
Software Consultant, ITT Avionics