“How do I know what I have to do?” -
The Role of the Inquiry Culture
in Requirements Communication
for Distributed Software Development Projects

Vesna Mikulovic
Siemens AG Austria, PSE
+43-51707-46138
vesna.mikulovic@siemens.com

Michael Heiss
Siemens AG Austria, PSE
+43-51707-46560
michael.heiss@siemens.com

ABSTRACT
As software specifications for complex systems are practically never 100% complete and consistent, the recipient of the specification needs domain knowledge in order to decide which parts of the system are specified clearly and which parts are specified ambiguously and thus need inquiry to achieve a more detailed specification. In this paper we classify 16 different situations (states) of requirements communication and analyze, based on a state diagram, how a mature inquiry culture can help to initiate transitions from undesirable states into more desirable states. In a case study the inquiry practices of a very large software development organization are shown. Knowledge networks within the organization play an important role in building up a mature inquiry culture.

Categories and Subject Descriptors

General Terms
Management, Documentation, Human Factors.

Keywords
Requirements communication, informal/formal communication, inquiry culture, requirement interaction management, global software development, state machine.

1. INTRODUCTION
Requirements engineering is one of the most significant factors having an impact on productivity in software development. The most common factors of project failure are linked directly to the quality of the requirements [13] and the maturity in requirements communication. It is rarely because of one or two reasons that IT projects fail, but rather because of unrealistic project goals, badly defined requirements and unmanaged risks, poor communication between customers and developers, as well as the inability to deal with the project’s complexity [3].

Copyright is held by the author/owner(s).
ICSE’06, May 20–28, 2006, Shanghai, China.
ACM 1-59593-085-X/06/0005.
the basis of structured interviews of these experts for a more detailed research on the key issues of the inquiry culture.

2. TERMINOLOGY

The implementation of complex distributed software development projects involves team members with a range of different skills and responsibilities. The individual team members have their own domain contexts and their own project contexts and therefore their own views on the specified system. The typical project structure includes about 30 project members and one project manager (PM), software engineering architects, developers and testers (Fig. 1).

![Figure 1: Requirement Communication takes place at and among all levels of the project organization](image)

The capability to communicate requirements without errors is an essential challenge in the requirements process. Requirements communication takes place at and among all levels of the project organization (Fig. 1). We call the one who knows about the specification the Sender S and the one who receives the specification the Receiver R. The upper end of each communication arrow in Fig. 1 points to the Sender, the lower end to the Receiver. The Sender S communicates the requirement specification to the Receiver R. E.g. the Sender S is the architect and the Receiver R is the developer.

One of the simplest software systems would be a system that has just one input \( x \) and one output \( y \). Every time the input \( x \) is entered into the system a predefined \( y \) is the expected output (Fig. 2). The requirement specification for such a system would be quite transparent: An input-output map like the one in Fig. 2 completely and consistently defines the system behavior. For each \( x \) exactly one \( y \) exists, which is the required output.

We use this transparent requirement visualization in our paper also for complex systems, because it still helps to understand the problems of requirements communication. In the case of a requirements specification of a complex system, the points of the input-output map are just abstract representations of requirements regarding the desired system behavior (Fig. 3). The values themselves have no numerical meaning but nevertheless show a deviation from a desired behavior and the necessity of “interpolating” incomplete or inconsistent requirements.

![Figure 2: Simple System: For every input \( x \) the required output \( y \) is clearly specified (completely and consistently)](image)

![Figure 3: Complex System: A set of requirements specify the desired system behavior only incompletely and sometimes inconsistently (Note that the third requirement shows an inconsistency). The combination of domain and context knowledge helps you to “interpolate” the missing details](image)

For each requirement the Sender S of the requirement may be in one of the four situations indicated below.

- The requirement is correct and \( S \) is convinced that it is correct.
- The requirement is correct but \( S \) is not sure that it is correct.
- \( S \) is not able to define it as final.
- The requirement is incorrect but \( S \) is convinced that it is correct.
- The requirement is incorrect and \( S \) knows that he is not convinced that it is correct.

On the other hand, Receiver R may have to deal with the same four situations for each requirement.

In order to help intuitive understanding, the abstract visualization of these four situations shows not only the requirement according to Fig. 3, but also the “absolute truth”, defined as the usually unknown complete set of requirements, which would optimize the system behavior of this specific project according to a certain optimization criterion. A full circle denotes that Sender S is convinced that his requirement is correct, while an empty circle denotes that Sender S is not sure whether the requirement is correct.

![Table 1: The 16 cases of requirements communication](image)

Note that we distinguish between the words correct and valid. A requirement can be officially considered as valid but nevertheless it is incorrect in terms of the objective absolute truth.
The combinations of these situations lead us to 16 cases of possible variations of requirements communication interaction between Sender/Receiver (Tab.1 and Fig. 4).

3. INQUIRY PRACTICES

Before we start to discuss the 16 different cases, we present some common practices of how inconsistencies and incompleteness are cleared up within Siemens Program and System Engineering PSE, a Siemens-in-house R&D division with 6200 engineers distributed over 20 sites.

3.1 Context

The first step for completing specifications or detecting inconsistencies is to apply personal context knowledge and common sense. If you have worked on previous projects with the same customer, you already have some context knowledge concerning this specific customer. By using common sense you can detect and solve obvious inconsistencies in your software requirement specification (e.g. requirement 3 in Fig. 3).

3.2 Domain Knowledge

The second step is to apply your personal domain knowledge. In a figural sense the combination of the context knowledge and the domain knowledge helps you to “interpolate” the details between the explicitly specified requirements (compare Fig. 2 and Fig. 3).

3.3 Inquiry within the Project Team (Peers)

If your personal context and domain knowledge is not sufficient or if you wish to get another opinion concerning your problem, you will ask other experienced colleagues in your project team.

3.4 Inquiry within the Organizational Knowledge Networks of the Receiver

A main asset of organizations like PSE is the compounded knowledge of all 6200 experts. An engineer of PSE may contact the proper PSE Knowledge Network [9], consult one of the internal consulting centers, i.e. PSE Support Centers [14], ask for resources from the PSE Know-How Base - either locally within his region or throughout the organization [11] - or he may post an Urgent Request to all members of all PSE Knowledge Networks [11]. In other words, the engineer has direct access to the knowledge of all experts within the organization and can therefore complement the knowledge he himself lacks efficiently and effectively.

3.5 Inquiry within the Project Hierarchy

The project manager tries to enable an open inquiry culture within his project organization. The art of the project manager is to build up an atmosphere in which every team member feels welcome to ask questions [8]. At the same time he “educates” the team members to use the peer networks of the whole organization for topics which can be more efficiently solved by them than by the project hierarchy. This also educates the team members to build up the competency to be able to ask mature questions.

3.6 Inquiry within the Knowledge Networks of the Sender

If you have worked on several projects for the same customer, you will usually have established personal networks with the colleagues of the Sender. This gives you the opportunity to get the missing project information through your informal networks.

3.7 Inquiry to the Sender

Depending on the type of missing or required information or the type of inconsistency, it can be the first and natural step to ask the Sender. In the first phases of the requirement definition it is desirable that the Sender and Receiver communicate intensively, because they have to speak in the same terminology. During the project the Sender usually expects a more specific communication and appreciates it if he is not involved in every small detail but only in topics that are relevant from his point of view.

3.8 Inquiry to External Sources

If neither the Sender’s organization nor the Receiver’s organization can provide the missing information, PSE may resort to the network of external sources it has built up and which can help in such cases [11].

![Figure 4: Visualization of a requirement space with 16 specified requirements corresponding to the 16 cases of Tab. 1. The upper graph shows the personal view of the Sender, the lower graph the personal view of the Receiver. Legend: ○: Requirement specification where the person is convinced that it is correct; ●: Requirement specification where the person is not convinced that it is correct; solid line: The unknown invisible “absolute truth”.

4. STRATEGIES FOR THE 16 CASES

Each of the 16 cases (Tab. 1) describes a different situation, which may occur when a requirement is communicated from the Sender to the Receiver. According to Fig. 3 and Section 2 these 16 cases are visualized in Fig. 4. The solid line shows the unknown absolute truth of the customer needs, as defined in Section 2.

Case 1: The requirement was correct and was correctly communicated. Hence, the Receiver R has the correct requirement and is convinced that this requirement is correct (Fig. 4, Requirement 1). No further action is required.

Case 2: The requirement was correct and was correctly communicated. Nevertheless, the Receiver is not convinced that this requirement is correct (Fig. 4, Requirement 2). It is the responsibility of R to decide whether some further action is required or whether, for reasons of efficiency, the (non-critical)
requirement is taken as it is. Usually, R tries to be convinced by involving his team colleagues or his personal networks as mentioned in Section 3.

**Case 3:** The requirement was correct, but the Receiver interpreted the requirement incorrectly. Nevertheless, R is (wrongly) convinced that the requirement is correct (Fig. 4, Requirement 3). Intensive communication with the Receiver enables the Sender to realize that the Receiver has misunderstood the specification and to clarify matters. An intact communication within the team enables R to realize that he has interpreted the requirement incorrectly (Fig. 5).

**Case 4:** The requirement was correct and the Receiver interpreted the requirement incorrectly, but he is not sure of his interpretation (Fig. 4, Requirement 4). The Receiver knows that he must act by using one of the inquiry practices of Section 3. The goal of this activity is to change the state to Case 2 or even better to Case 1 (Fig. 5).

**Case 5:** The requirement is correct, but the Sender is not convinced that it is correct. Nevertheless, the Receiver understands the requirement correctly and is convinced that the requirement is correct (Fig. 4, Requirement 5). Due to his context and/or domain knowledge he was able to ascertain the correctness of the requirement. In the course of communication the Sender may be convinced as well (Fig. 5).

**Case 6:** The requirement is correct but both the Sender and the Receiver are not convinced that the requirement is correct. As both have a different context and knowledge, the missing information can be added by discussion to make sure that both parties are convinced in the end. (Fig.5). Otherwise, both the Sender and the Receiver may offer to involve their networks.

**Case 7:** The requirement was correct, but the Sender is not convinced that it is correct. The Receiver interprets the requirement incorrectly. Nevertheless, R is (wrongly) convinced that the requirement is correct (Fig. 4, Requirement 3). The strategy is similar to Case 3. The advantage of Case 7 is that the uncertainty of the Sender may trigger a clarifying discussion (Fig. 5).

**Case 8:** The requirement was correct, but the Sender is not convinced that it is correct. The Receiver interpreted the requirement incorrectly but he is not convinced of his interpretation (Fig. 4, Requirement 8). Like in Case 7 the Receiver will implement a wrong requirement if no action is taken. Therefore the Receiver uses one of the inquiry practices of Section 3 (Fig. 5).

**Case 9:** The requirement is incorrect, but the Sender is (wrongly) convinced that it is correct. The Receiver corrects the requirement and is convinced that this is what the Sender really meant. It is the responsibility of the Receiver to decide whether it is necessary to convince the Sender or whether it is an uncritical requirement that does not need to be corrected (Fig. 5).

**Case 10:** This case is similar to case 9 except that the Receiver is not convinced that his proposed requirement is correct.

**Case 11:** The requirement is incorrect, but the Sender is (wrongly) convinced that it is correct. The Receiver takes the requirement as it is and is therefore wrongly convinced that the requirement is correct (Fig. 4, Requirement 11). This is the most dangerous situation between the Sender and the Receiver, as neither can see the problem. If both have an adequate communication culture within their teams, there is the chance that one of them recognizes the mistake (Fig. 5).

**Case 12:** The requirement is incorrect, but the Sender is (wrongly) convinced that it is correct. The Receiver understands the requirement as it was communicated to him, but he is not convinced that it is correct (Fig. 4, Requirement 12). The Receiver is the only one who can change the state (Fig. 5), using the methods described in Section 3.

**Case 13:** This case is similar to Case 9. The requirement is incorrect, and the Sender is not convinced that it is correct. The Receiver corrects the requirement and is convinced (Fig. 4, Requirement 13) that his version of the requirement is what the Sender really meant (Fig. 5).

**Case 14:** The requirement is incorrect and the Sender is not convinced that it is correct (he feels that something is wrong). The Receiver has corrected the requirement, but he is not convinced that his proposed requirement is what the Sender really meant (Fig. 4, Requirement 14). Again, the expertise of Sender and Receiver can help to clear up this case (Fig. 5). Moreover, both the Sender and the Receiver may involve their networks to get additional information and views.

**Case 15:** The requirement is incorrect and the Sender is not convinced that it is correct (he feels that something is wrong). If, for some reason or other, the Sender tries to hide his uncertainty, the Receiver might take the requirement to be correct and, therefore, will implement this wrong requirement if no action is taken. As R is (wrongly) convinced that the requirement is correct, he will not be the one to initiate any action (Fig. 4, Requirement 15). Although it is the responsibility of the Sender to inform the Receiver of his doubt, an intact communication within the team of the Receiver can help to recognize the mistake (Fig. 5). However, an intact partnership between Sender and Receiver would provide an open communication between them and prevent information hiding.

**Case 16:** The requirement is incorrect, and the Sender is not convinced that it is correct (he feels again that something is wrong). The Receiver is not able to correct the requirement on his own, but at least he is not convinced either that the requirement is correct (Fig. 4, Requirement 16). This is a dangerous situation for the project, if none of the two parties reacts properly. Nevertheless, Case 16 is less dangerous than Case 11, as both are aware of their uncertainty. Again, the different views of Sender and Receiver can help to clarify this case. If not, both the Sender and the Receiver may involve their networks for additional support and thus arrive at a more desirable state (Fig. 5).

### 4.1 State Transitions

The description of these 16 cases presents all possible situations between the Sender and Receiver. Without loss of generality we assumed that the set of requirements remain constant during the project. In other words, the continuous line in Fig. 4 and the continuous line of the state visualizations in Fig.5, called absolute truth remains constant. This is not the case in all projects. Note that a change of the absolute truth without the knowledge of Sender and Receiver is an unrecognized transition from State 1 to State 11. As explained in Case 11, it is much more likely to detect
such a situation if at least one of the two parties, either the Sender or the Receiver, recognizes this situation (State 12 or State 15). In all 3 cases a mature communication and inquiry culture helps to avoid severe problems for the project. The most dangerous cases occur if no communication action is taken (Fig. 5).

Note that some states need a longer path for transition to a highly desirable state: e.g.: a typical path from State 16 to State 1 would follow the states $16 \rightarrow 14 \rightarrow 13 \rightarrow 5 \rightarrow 1$.

Figure 5: State diagram of the 16 requirements communication cases (states) according to Tab. 1. Without a proper inquiry culture 50% of the states remain in an undesirable state. Legend: $\rightarrow$ : state transition by an inquiry; $\Rightarrow$ : state transition which may cause severe problems for the project; $\ldots\ldots\ldots\ldots\ldots\ldots\ldots$ : other state transitions. (For details see the case description in Section 4.)

5. CONCLUSION

In this paper we have presented the virtual requirements communication state machine which, in a mature customer-developer partnership usually operates on an unconscious level. The goal of this research is to get more insight into, as well as reproducibility and consciousness of these informal communication-based requirements techniques.

Today professional software development without processes is unimaginable, but also software development without integration of informal requirements techniques will result in severe efficiency problems. The goal of our work is to integrate the inquiry culture into the software development processes and to find ways for stimulating the inquiry culture until it becomes a living culture in organizations.

We demonstrated that a missing inquiry culture may cause the implementation of requirements which do not satisfy the customer’s needs. It will require additional money, time, and resources to mend matters, and it will lead to a loss of reputation.

Until now our research has been based on informal interviews within PSE organizations in 9 different countries. Based on the presented classification, we plan structured interviews for a more detailed insight into and a more validated knowledge of the key issues of inquiry culture.

Open issues: We assume the existence of an “absolute truth”. We state that the absolute truth is neither known by the Sender nor by the Receiver, and it is even imaginable that it does not exist objectively. One reason could be that both the Sender’s view and the Receiver’s view equally fulfill the optimization criterion.

Is it conceivable that, during a project, the Sender and the Receiver classify each requirement according to the 16 cases of Fig. 5? In this case they cannot classify it based on the absolute truth but only on their personal opinion of what this truth is. Up to now we have not analyzed whether this approach is practicable and has enough impact on a project.

6. REFERENCES


