Before we start in defining the meaning of each feedback type, we would like to state that there are two different kinds of feedback, a ***simple feedback***, and a ***complex feedback***. A simple feedback is a feedback that consists of a single feedback type that a user provides in his post to express a certain meaning, while the complex feedback is a structured feedback that consists of several feedback types that together form a new meaning that can be inferred from its unique structure. Below if the list of feedback types and subtypes (i.e. cases):

**1. Confirmation or Negation:** is a simple feedback type that the users use to agree or disagree on problems or opinions of other users. When these feedback types are unaccompanied with other types in a feedback, it can be inferred as voting for a problem or a given solution.

**2. Investigation** is a simple feedback type used when a user is asking a question to clarify something about another feedback posted by another user. A user may ask about some issues in a problem statement, or unclear steps in a provided solution, or clarify some contextual information that helps explain the problem more.

**3. Elaboration** is a simple feedback type where the user gives extra explanation on a feedback he already posted. There are two cases for giving extra explanations on a feedback:

**a. Feedback Elaboration:** is when a user needs to give more detailed information that he forgot to provide in his main feedback this can be added separately in the feedback where he elaborates. For Example, A user can elaborate on a problem he provided by giving explanation on some trials that he made trying to solve his problem or rephrasing the problem statement.

**b. Investigation Elaboration:** is when a user simply replies on an Investigation by giving detailed explanations to answer the posted question(s).

**4. Justification** is a simple feedback type used when users need to provide reasons to support their feedback. They may give reasons why they provided a solution/ suggestion, or it can be used with confirmations or negations to state reasons why a user agrees or disagrees on a feedback opinion of another user.

**5. Verification** is a complex feedback type where a user gives his opinion on a solution or suggestion he received on the problem that he posted. As a complex type it means that it combines several other feedback types in its structure that are mandatory in its definition. Specifically in order to verify whether a solution or a suggestion was useful or not, this feedback has to reference a certain Mitigation (i.e. Solution or Suggestion) in which the user will be giving his opinion to verify whether it solved the issue or not by using Confirmation or Negation.

**6. Problem** feedback type refers to a certain feature or group of features in the software that the user is having problem with, and a detailed explanation of the problem. Problems may use other feedback types such as Investigations to ask users some questions they need answers for. However, problems in general cannot occur in the same Feedback post with Mitigations or Verifications. In general users who post problems are not the same users who post the Mitigations, and even if this case occurred will not be contained in the same problem post.

**a. Topic definition** is a simple feedback type that represents the first posted problem in a feedback thread where the user is seeking help. Therefore it does not reference any other feedback in the thread but can be referenced in many other posts.

**b. Addition** is a complex feedback type where a user votes (i.e. agrees or disagrees) on any posted problem, and adds another problem in his feedback, which is not related to the main problem on which the discussion is held.

**c. Problem Extension** This is a complex feedback type where a user tried a Mitigation and it solved part of the problem, BUT led to another related problem to occur.

**d. Mitigation Trial Failure** This is a complex feedback type, where a user confirms on a posted problem (i.e. he has the same problem), AND tried the mitigation that was posted by other users in attempt to resolve the problem, BUT couldn't try the mitigation (so this is a new problem for him besides the main one).

**7. Mitigation** is a complex feedback type that represents a solution or a suggestion that may help a user resolve the problem(s) he has. Since this type is intended to resolve a problem, therefore it has to reference that problem in the solution or suggestion for specificity. Also, for every Mitigation it is always expected that the user who posted the problem will Verify that Mitigation. There are two types of Mitigations:

**a. Solution** is a well-known procedure or steps that when followed can resolve the problem or issue.

**b. Suggestion** is a recommendation that a user provides for another user as a trial to resolve his problem. This suggestion may or may not solve the problem. This needs Verification from the problem owner (i.e. the user who posted the problem).

**8. Correction** is used when a user corrects the understanding of another user. There are two cases for this feedback type.

**a. Problem correction** is a complex feedback type. It occurs when the user corrects the problem of another user. In a problem definition a user must refer to a feature(s) that he is having a problem with. Sometimes the user is using a feature which is not intended for the type of task he is doing, simply due to a lack of understanding of the job a feature should perform. Consequently, other users can provide corrections to this misunderstanding.

**b. Mitigation Correction** is a ***complex*** feedback type. This type of feedback may occur when a user is trying to correct a Mitigation that was provided for a certain problem. Errors in Mitigations may occur due to the lack of contextual information about the tasks the user is doing or environmental information about the softwares or hardware used while applying Mitigation.

Below is **the novel description of the Detail Types**:

**1. Concise.** By literal meaning it is used when users provide very short feedback types with no explanations or details. From the analysis we noticed that it is used mostly, when users tend to confirm or negate by just expressing their agreement or disagreement on a feedback. Moreover, it was never used in problem statements or mitigations, since by nature these specific feedback types need explanation to be meaningful.

**2. Explanation** is the opposite of concise, as in this detail type the user is expected to provide as much details in his feedback to make it meaningful for other users. There is no restriction on the use of this detail category with any feedback type, because it is always acceptable to give more details.

**3. Exemplification** is utilized when the users need to provide examples within this text. In the forums’ threads that we have analysed examples are always given within explanations especially problem explanations.

**4. Trials** is used closely with problem description where the problem owner who is explaining the problem, shows that he made many attempts to resolve the problem but have failed to reach a Solution. The user posts these trials as a kind of extra explanation of the problem and how it occurs, and also to avoid getting suggestions from other users with same trials that he already made.

**5. Scenario** is used to explain text in a list. A solution can be explained in steps. These steps if verified by the problem owner can be used as a solution scenario to solve similar problems to other users. Moreover, other users may list the problems they have in the problems statement. Other may suggest mitigation to other users in a form of a list of possible actions to try; sometimes it matters to be in a certain order.

**6. Feature Definition** is used to define a user’s perception of the usage of a certain feature. This description is sometimes used in problem statements, which helps other users understand why the user is having a problem (i.e. sometimes users have wrong understanding of the usages of a feature). Moreover, users who provide Mitigation may use it a form to document how they use a feature with certain types of tasks. Finally, it is mostly used when users provide Feedback Type: Correction, specifically Problem Correction, where the user corrects the misunderstanding of another user by providing the correct feature definitions to features referenced in the problem statement.

**7. Question** is a simple detail category that is used with Investigations to indicate the question(s) posted for clarification.

Contextual information can carry valuable information that can help make the feedback more understandable or useful. There are five main categories of contextual information that were captured in the forums analysis that map to [13].

**1. Task:** It captures what the user is doing. This is specifically important when the user is describing a Problem feedback type, because it gives to the other users an idea about the context in which the problem occurred, or describing the frequent jobs that the user is involved in in his daily work which helps give an idea to other users about the importance the feature the user is having problem with.

**2. Spatio-Temporal:** In this kind of context the user specifies information related to place and time. From our forums analysis we have found an angle where such information may play useful role. Cases are when users try to explain the timing relationship between two tasks (i.e. two tasks happening together, or one feature corrupts when a user does a certain action). Another Case is when users try to specify some information about a problem in relation to where it occurs in software for example in a certain interface, or when using a certain module.

**3. Personal:** In this kind of context users express their emotional judgments, stress, or information about their expertise, which is repeated mainly with Negation feedbacks.

**4. Social:** we mean context information related to a user’s role at work, and information about co-workers.

**5. Environmental:** is related to a software or hardware specs, versions, and architectures. Users can provide these kinds of information in a problem statement to specify the software version they are using which may differ in the feature with problem from older or newer ones. Therefore, this adds specificity and usefulness to add such information. Moreover, users can add also environmental context in Mitigations to specify that the suggestion or solution works on a certain version, or works well with a certain hardware configuration.