The main points of failure are:

1. NOT PAYING ATTENTION TO YOUR FEEDBACK ON PREVIOUS DELIVERABLE, particularly the requirement to hand in a revised earlier deliverable. This failing earns a 0. I want the group to learn that NOT listening to YOU is a BIG NO NO now, when the points don't count much.

Nevertheless, give whatever feedback you can on the deliverable they did hand in.

This deliverable is probably the hardest to mark, because marking it correctly requires understanding what the UCs are and what they do. However, I believe that by this time, you have this understanding of your teams' projects.

Based on past experience, I would say that many will give good lists of assumptions, exceptions, and variants. These will be largely at the level of the UCM, and this is a GOOD thing. While not deprecating these lists, many will not list many of the lower-level, data-level exceptions. Also, I see that many do not have at least one exception for every assumption.

So the principal failings are:

1. Not having at least one exception for every assumption. At the very least, the negation of an assumption is an exception. Sometimes, there is not ONE exception to an assumption; there are multiple exceptions, the logical disjunction of which is the logical negation of the assumption. :-)

For any such exception, the group may decide in the end that its system may not deal at all with the exception, that if the assumption does not hold during the running of the system, the system will fail. This treatment will usually be that the exception is not detectable or even if detectable, there is nothing the system can do about it. But then, the assumption, the exception, and

the decision not to handle it will need to be documented in the final specs. Now is too early to make this decision. So all we expect for this deliverable is the assumption and the exception.

2. Ignoring the common, everyday mistakes that a user can make on input, including just plain clicking on the wrong buttons or typing the wrong kind of text, e.g., a letter when digits are expected, etc. While these exceptions are not exciting, not difficult to deal with, etc., dealing with these ends up being a majority of the code that needs to be written.

If there is ANYTHING that they have done that you do not understand, comment on that. Say that this is not clear. Ask them what they mean. Sometimes, lack of clarity to you is a sign of something that is unclear to them. Regardless, it is their job to be clear.

With regard to variants, almost anything is fine. The purpose of having them list variants was just to get them to think about them. These will generally be either fairly obvious or fairly innovative. If they have NONE, ask, "Where are the variants?" If you think of a variant that they did not, you can ask about it.

Of course, if the UCM of D2 was or is still deficient, then any list of assumptions, exceptions and variations will be deficient. So the team may have to go back and redo their UCMs again.