Reading Technique for Use Case Construction

Create use cases to document the functionality that users of the system should be able to perform. This will require listing the functionality that the new system will provide and the external interfaces that are involved. You will have to identify actors (sets of users) who will use the system for specific types of functionality. Remember to include all of the functionality in the system, including special/contingency conditions. Follow the procedure below to generate the use cases, using the questions provided to identify faults in the requirements:

1) Read through the requirements once, finding participants involved.

    a) Identify the participants in the new requirements. Participants are the other systems and the users that interact with the system described in the requirements – that is, they participate in the system functionality by sending messages to the system and/or receiving information and instructions from it. List these participants on Form A. You may use the following questions to help you identify participants:
        - Which user groups use the system to perform a task?
        - Which user groups are needed by the system in order to perform its functions? These functions could be its major functions, or secondary functions such as system maintenance and administration.
        - Which are the external systems that use the system in order to perform a task?
        - Which are the external systems that are managed or otherwise used by the system in order to perform a task?
        - Which are the external systems or user groups that send information to the system?
        - Which are the external systems or user groups that receive information from the system?

    Q1.1 Are multiple terms used to describe the same participant in the requirements?
    Q1.2 Is the description of how the system interacts with a participant inconsistent with the description of the participant? Are the requirements unclear or inconsistent about this interaction?
    Q1.3 Have necessary participants been omitted? That is, does the system need to interact with another system, a piece of hardware, or a type of user that is not described?
    Q1.4 Is an external system or a class of "users" described in the requirements which does not actually interact with the system?

2) Read through the requirements a second time, identifying the product functions.

    a) Identify the set of functionality that the system must be able to perform. That is, what activities do the requirements say the system must do? (E.g. display a database record, print a report, create and display a graph.) Record the system functionality on Form A.
    b) Now consider how a user of the system will view it. He or she is probably not concerned with the individual activities that the system can perform, but instead thinks about using the system to achieve some goal (e.g. adding information to an existing database, computing a payment, viewing the current status of an account). These user goals will be the set of use cases for the system.
    c) For each use case, use a new copy of Form B. Decide which of the system activities recorded on Form A are involved in this use case, and note them on Form B. Sketch out, from the user's point of view, what steps are required to achieve the goal and use arrows to identify the flow of system control ("First this functionality happens, then this…"). Use branching arrows to signify places where the flow of control could split. Remember to include exception functionality in the use case. Check the appropriate functional requirements to ensure that you have the details of the processing and flow of control correct.
    d) For each use case you create, remember to signify what class(es) of users would use the functionality (the user classes should be already recorded in the list of participants on Form A), as well as the action that starts the functionality (e.g. "selecting option 2 from the main menu" might start a use case for deleting records from a database). There are spaces for recording both of these

pieces of information on Form B. Finally, give the use case a descriptive name that conveys the functionality it represents and assign it a number for future reference.

Q2.1 Are the start conditions for each use case specified at an appropriate level of detail?
Q2.2 Are the class(es) of users who use the functionality described, and are these classes correct and appropriate?
Q2.3 Is there any system functionality that should be included in a use case but is described in insufficient detail or omitted from the requirements?
Q2.4 Has the system been described sufficiently so that you understand what activities are required for the user to achieve the goal of a use case? Does this combination of activities make sense, based on the general description and your domain knowledge? Does the description allow more than one interpretation as to how the system achieves this goal?
Q2.5 Do the requirements omit use cases that you feel are necessary, according to your domain knowledge or the general description?

3) Match the participants to all of the use cases in which they are involved. (Remember that if two participants are involved in all of the same use cases, they might represent a single unique actor and should be combined.) Use the column marked "Involved in which use cases?" on Form A to cross-reference the participants with the appropriate use cases.

Q3.1 Is it clear from the requirements which participants are involved in which use cases?
Q3.2 Based on the general requirements and your knowledge of the domain, has each participant been connected to all of the relevant use cases?
Q3.3 Are participants involved in use cases that are incompatible with the participant's description?
Q3.4 Have necessary participants been omitted (e.g., are there use cases which require information that cannot be obtained from any source described in the requirements)?

Acknowledgements
These guidelines owe much to the following works:

Andersson, Michael, and Johan Bergstrand. (1995) *Formalizing Use Cases with Message Sequence Charts*. Masters Thesis for the Department of Communication Systems at Lund Institute of Technology.
Jacobson, Ivar, *et al.* (1992) *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Publishing Company.

And to the work of Curt Zimmerman and Guilherme Travassos during their time with the Empirical Software Engineering Group.