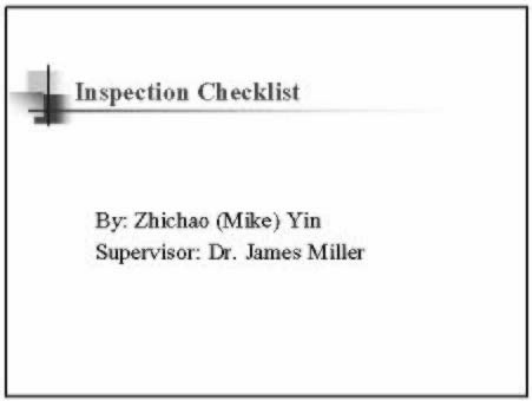
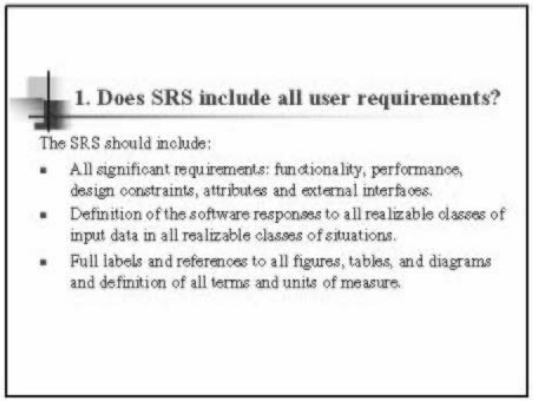
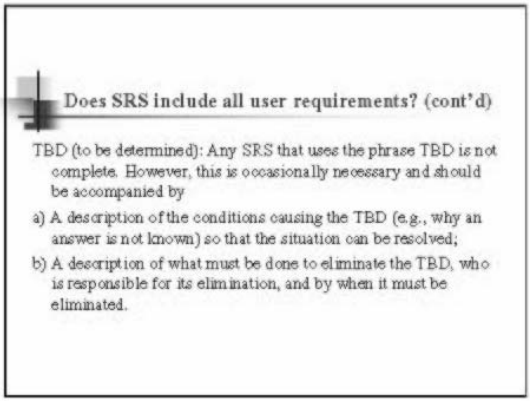
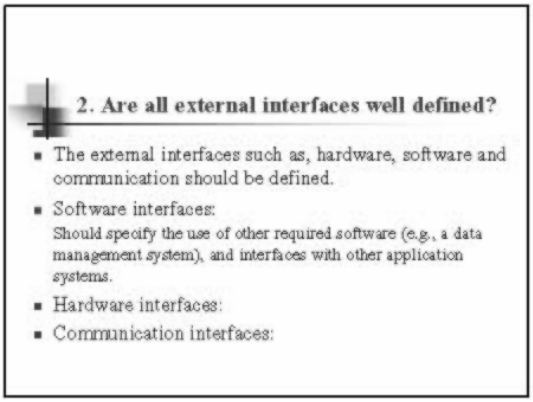
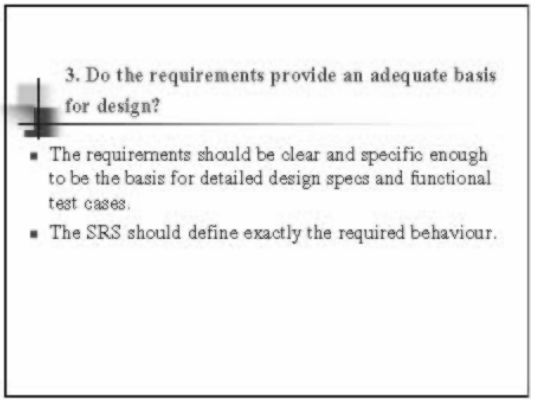
Appendix B. Inspection Checklist











4. Are algorithms intrinsic to the functional requirements defined?

- Example:

Function Requirement 1:

1. Description: List the users in terms of the ascending frequency of logging into the sub-system.
2. Input: …
3. Processing: …
4. Output: …
5. Algorithms: Bubble sorting algorithm.

5. The expected behavior is documented for all error conditions?

- The requirements should Respond to abnormal situations.
- Example:
  When the user inputs an invalid (lengthy) pin, the system should prompt as "Please input the correct PIN!"

6. All the necessary objects in requirement are well defined?

- These objects include the data, format of data, and function etc.
- Example:
  - Customer account number is defined as integer.
  - The default value for maximum number of concurrent transactions is 2,000.

7. Do any requirements conflict with or duplicate other requirements?

- The SRS should agree with any applicable higher-level specification,
- No internal inconsistency between the requirements
  The specified characteristics of real-world objects may conflict.
- Duplication.

8. Are all internal references to other requirements correct?

- References amongst the requirements conflict.
- Such as:
  The maximum number of items permitted on rental = the maximum number of items that can be rented in a particular transaction.

9. Are all requirements written at an appropriate level of detail?

- The SRS should contain necessary implementation detail to avoid under-specified functional requirement.
- Avoid the unnecessary details, which will lead to over-specification.

10. Is each requirement written in standard terminology and definitions?

- The requirement should be written in a clear, concise and unambiguous language. (IEEE std 1074-1997)
- The SRS should be unambiguous both to those who create it and to those who use it.

11. Is each requirement verifiable by testing, demonstration, review, or analysis?

- It must be possible to develop a through set of tests based on the information contained in the SRS.
- Example:
  Non-verifiable requirements include statements such as "works well", "good human interface", and "shall usually happen" etc.

12. Is each requirement free from errors in content?

- The error will lead to an ambiguous understanding,
- Example:

  After inputting the user name, the system should prompt for "password" to login to the system. (Right)

  After inputting the user name, the system should prompt for "other identifications" to login to the system (wrong).

13. Can all of the requirements be implemented within known constraints?

- Sketch a specific test that establishes under the known constraints to test the satisfaction of requirement.
- Example: ABC video renting system.

  Constraints:
  - All customers and rental items must be registered with the system.
  - For every rental item, there must be an applicable fee schedule.
  - Customers must have a valid credit card or be able to make a deposit in cash or by check in order to register.

  The functional requirement is: the clerk should be able to register a customer.

14. Are all performance objectives properly specified?

95% of transactions can be finished in 1 second instead of "most of transactions can be finished in a short time"

15. Are all security and safety considerations specified in sufficient detail?

- Keep certain functions in separate modules;
- Permit only limited communication between some areas of the program;
- Check data integrity for critical variables.

16. Is the SRS traceable?

- Backward traceability:
  - to previous stages of development. This depends on each requirement clearly referencing to its source in earlier documents.
- Forward traceability:
  - to design, code and test document. This depends on each requirement in the SRS having a unique name or reference number.

17. Are all requirements actually requirements?

Are all requirements actually requirements, not design or implementation solutions?

- Since the SRS has a specific role to play in the software development process:
  1. Should not describe any design or implementation details.
  2. Should not impose implementations on the software.

Note:

- Don't limit to these items.

References:

1. IEEE, IEEE Recommended Practice for Software Requirements Specifications – IEEE Std 830-1998
2. Stuart R. Faulk, Software requirements: A tutorial, NRL report, Naval Research Lab, Wash. DC., 1995
3. Ian Sommerville, Software Engineering, 6th Edition, Part 2: Requirement, Addison-Wesley