# Design Review Topics and Suggestions

## *Aim of Review*

The aim of a design review is to ensure

- Requirements satisfied
- Test Suite can be designed with design document input
- Coding can be done from design documents
-

To these ends it is suggested that the reviewers receive:

1) A general explanation of the purpose of the software. (Description of inputs, outputs, functions, constraints, list of components, source documents). This could usefully be the first section of the Architecture document.

2) Architectural document
This includes diagrams and explanatory text.
   a) Major classes
   b) Inheritance
   c) Aggregation (containment)
   d) Interfaces
3) Detailed design document
   a) classes in detail
   b) attributes
   c) methods
   d) description of behaviour of methods
4) Data Architecture document (Also called the Data view of the design)
   a) major data structures
   b) data base use design (tables, data types)
5) Requirements traceability
   a) matrix showing between software component and use case
   b) and/or Collaboration or sequence diagram illustrating use cases (design walk through of usecases)
6) Other salient features to which the the authors would like to draw attention
   Examples: Testing, QA plans

Notes:
Attention should be paid to:
- sufficient visibility: Can objects can access what they need
- necessary visibility: Is the object's visibility really necessary?
- organise for reuse
- minimise complexity - ease maintenance, diminish coding load.

-------------------------------------------------------------------

Design Review will check for:
Is the overall explanation understandable?
Is there there anything ambiguous?
Is there anything missing (all use case/requirements satisfied)?
Also consider
- Ease of use of components
- Portability
- Maintainability
- Re-usability
- Performance

Detail

Careful examination of UML diagrams for all classes.

- interplay among interfaces and concrete classes
- parameters of templated classes
- the sense of the inheritance structures
- do subclasses provide appropriate implementations of :
    - overridden methods
    - overloaded methods?
    - hiding?
- compatibility among components, i.e.make sure that the signatures match up ( *This means that the method names and signatures appear in the class diagrams not an inspection of the cxx files)*
- destruction of owned components should be part of the design
- do components have access to all they need to satisfy the their requirements?