2.2) Specification Languages (SL) describes a system diagrammatically, in terms of: - Nodes => functions (nodes B,C might nest in ‘A’, for example); - Arcs => data flows. Checks for self-consistency might include:

-> do the functions (which ===data transformation) always maintain the same invariant(s) and when nesting is illustrated, do the nested transformations facilititate/permit adhering to invariant of nestee? (ie: functionA(int a , int b) { return (double?) functionB ( a , b ) {

return double } }

* Are null flows allowed/disallowed consistently?
* Can functions requiring/forbidding data/null receive their counterparts consistently OR shows ‘throwing a flag’ consistently?

2.4) Comparing relative approaches of (4) methods to ensuring serializability based on (3) axes of verification techniques (A) – Pessimistic Inaccuracy; (B) – Optimistic Inaccuracy; (C) Simplified Properties…..

(1): A,B,C are the same essentially as A,B are ==0 (there is no inaccuracy) due to C==true

(2) A is the case here since the analysis may reject some valid sets. B is not the case as the analysis should not give false positives. Automated static analysis cannot prove logical correctness therefore C is false.

(3) A,B should be false as conformance to a specification would seem a simplified-property check (C==true); however, if the