Abstract
ET: Enhanced tool (solution strategy)
The effectiveness of documenting a system in supporting the design of software has been demonstrated. An enhanced tool / method is described for the design of software based on documenting requirements in a modularly decomposed fashion. Examples are provided confirming the effectiveness of its support for software development in design.

Why ET?
The paper proposes what a requirements document should look like, both ideally and as a practical result of a real development process. Given that this paper was written in the mid-1980’s, it is hard to believe that the sort of requirement document he proposes is too “radical” compared to previous projects’ documentation—even Brooks argues for the importance of this sort of documentation in MMM—but the author would have us believe that his contribution is a significant enhancement on the state-of-practice in documentation. It is not an experience report, since though it presents heuristics implicitly, it is not based directly on an analysis of existing experience or past projects.

Question - [Method/means of development]
<model-type / solution strategy>
What is the most helpful way of documenting requirements? What is the “ideal” documentation strategy, and why is it worth pursuing, to the extent we are able?

<artifact-type>
The artifact being produced is a software implementation. The question is how documentation can support the goal of providing correct, maintainable software.

Results - [Procedure / technique]
<model-type / solution strategy>
The authors outline what a requirements document should look like. Interestingly, they propose that this document should be architected along a modular decomposition (with all the connotations that these words usually carry in the context of software design!)
• Use interfaces to separate concerns
• Assign concerns to respective modules
• Document the structural hierarchy of those modules
• Document the interfaces of those modules
• Document the internal structures of those modules (which is a somewhat questionable thing to propose at the requirements stage, since it gently tap-dances around in that grey area between specification and implementation)
• Document the inter-module references

There should be a particular emphasis on cleanly documenting issues likely to receive change orders in the future. The goal is to codify the reasons behind designers’ decisions.

The authors address the fact that it’s practically infeasible to achieve an ideal document on the first go. However, they don’t seem to go so far as mandating that we “plan to throw one away.”

Validation – [Persuasion]
<model-type / solution strategy>
Although the authors resort to “blatant assertion” in a few places, they do offer a handful of reasons why this approach is right. Most reasons rest on the day-to-day challenges of practical software development (such as the fact that staff turnover necessitates a record of how requirements led to the software’s design).