
Type of paper: This paper is an experience report.

Correlation to the real-world setting: The paper reaches its conclusions by generalizing from several successful and unsuccessful real-world systems.

What is the hypothesis: Principles of interfaces derived by studying prior systems may prove useful in creating future systems.

Validation: None is offered in the paper. Evaluation will be difficult because some of the principles contradict. However, the author does provide examples of previous systems that illustrate the principle.

Result: A long list of interrelated principles. Broadly: keep it simple (but not too simple), optimize interfaces for speed, keep continuity in the interfaces, avoid generalization, handle best and worst case scenarios separately, failure is an option in some cases, do end-to-end checks, use logging, and make actions atomic.

Do you believe the result, and why? Some of them I believe others I question. I was involved in at least two networking projects that attempted to move TCP checksums into hardware. On both projects we discovered data corruption. I think that calling interfaces a programming language is an overstatement. In my experience, it takes less time to design an implement an interface than a programming language and compiler. Plan to throw one away is an overstatement: I developed several systems are rarely needed to throw the first system away. However, this did happen in at least one case.