
Abstract
XH: Experience and/or Heuristic
Studies reported here of <a variety of extant systems, principally operating systems,> supported by <various design approaches> generate a number of findings concerning <how to effectively design computer systems>, including <a variety of design heuristics>. They indicate that <proper, efficient, fault-tolerant function> is (usually) met by <ensuring that the system always works properly, perhaps by taking advantage of fault-tolerance techniques, and then optimizing for the common case>.

Why XH?
The paper is largely a review of lessons to be drawn from systems that have already been built, both by the author and by other people. As such, it is a resume of various experiences and the heuristics learn from those experiences.

Question - [Method/means of development]
<issues>
What is the most effective way to design computer systems?

<requirement>
Main criteria: ensuring that systems work properly and efficiently, perhaps amid faults.

Results - [Procedure / technique]
<list-of-findings>
The authors propose the following variety of design heuristics...

<design-heuristic>
• Separate the normal case from the worst case
• Make the normal case fast
  o Focus on doing one thing well
  o Let the client do work if necessary
  o Keep interfaces simple
  o Use static analysis when feasible
  o Compute in batches or in the background
• And make all cases correct
  o Save a small kitty of resources that you can throw at worst cases
  o Provide end-to-end correctness checks
  o Facilitate fault-tolerance through atomicity, idempotency, and logging

Validation – [Experience]
<application>
The authors cite a number of extant systems (principally operating systems) successfully incorporating these design heuristics. They also mention a handful of cases where the design heuristics were not followed, to the woe of their designers and users.

<supporting technology>
As noted above, not all of these extant systems demonstrated these design heuristics.