Principles of Computer System Design

An Introduction

Design Principles

Jerome H. Saltzer

M. Frans Kaashoek

Massachusetts Institute of Technology

Version 5.0

Suggestions, Comments, Corrections, and Requests to waive license restrictions: Please send correspondence by electronic mail to:

Saltzer@mit.edu

and

kaashoek@mit.edu

Computer System Design Principles

Throughout the text, the description of a design principle presents its name in a **bold-faced** display, and each place that the principle is used highlights it in *underlined italics*.

Design principles applicable to many areas of computer systems

- **Adopt sweeping simplifications**
  So you can see what you are doing.
- **Avoid excessive generality**
  If it is good for everything, it is good for nothing.
- **Avoid rarely used components**
  Deterioration and corruption accumulate unnoticed—until the next use.
- **Be explicit**
  Get all of the assumptions out on the table.
- **Decouple modules with indirection**
  Indirection supports replaceability.
- **Design for iteration**
  You won't get it right the first time, so make it easy to change.
- **End-to-end argument**
  The application knows best.
- **Escalating complexity principle**
  Adding a feature increases complexity out of proportion.
- **Incommensurate scaling rule**
  Changing a parameter by a factor of ten requires a new design.
- **Keep digging principle**
  Complex systems fail for complex reasons.
- **Law of diminishing returns**
  The more one improves some measure of goodness, the more effort the next improvement will require.
- **Open design principle**
  Let anyone comment on the design; you need all the help you can get.
- **Principle of least astonishment**
  People are part of the system. Choose interfaces that match the user's experience,

expectations, and mental models.

- **Robustness principle**
  Be tolerant of inputs, strict on outputs.
- **Safety margin principle**
  Keep track of the distance to the edge of the cliff or you may fall over the edge.
- **Unyielding foundations rule**
  It is easier to change a module than to change the modularity.

Design principles applicable to specific areas of computer systems

- *Atomicity:* **Golden rule of atomicity**
  Never modify the only copy!
- *Coordination:* **One-writer principle**
  If each variable has only one writer, coordination is simpler.
- *Durability:* **The durability mantra**
  Multiple copies, widely separated and independently administered.
- *Security:* **Minimize secrets**
  Because they probably won't remain secret for long.
- *Security:* **Complete mediation**
  Check every operation for authenticity, integrity, and authorization.
- *Security:* **Fail-safe defaults**
  Most users won't change them, so set defaults to do something safe.
- *Security:* **Least privilege principle**
  Don't store lunch in the safe with the jewels.
- *Security:* **Economy of mechanism**
  The less there is, the more likely you will get it right.
- *Security:* **Minimize common mechanism**
  Shared mechanisms provide unwanted communication paths.

Design Hints (useful but not as compelling as design principles)

- **Exploit brute force**
- **Instead of reducing latency, hide it**
- **Optimize for the common case**
- **Separate mechanism from policy**