Hints and Principles for Computer System Design

Butler Lampson
Microsoft Research
MSRA Faculty Summit
October 30, 2014
Overview

A 30-year update of my 1983 *Hints for Computer Systems*

These are hints, often not consistent or precise

Just a few principles

Hints *suggest*, principles *demand*

— No nitpicking allowed

**STEADY by AID**

What: Simple, Timely, Efficient, Adaptable, Dependable, Yummy

How: Approximate, Incremental, Divide & conquer, …

*There are three rules for writing a novel. Unfortunately, no one knows what they are.*

— Somerset Maugham

*You got to be careful if you don’t know where you’re going, because you might not get there.*

— Yogi Berra

*The quest for precision, in words or concepts or meanings, is a wild goose chase.*

— Karl Popper
What: Goals

Simple
Timely (to market)*
Efficient
Adaptable*
Dependable
Yummy*

*More important today

Need tradeoffs—You can’t get all these good things

[Data is not information, ] Information is not knowledge, Knowledge is not wisdom, Wisdom is not truth, Truth is not beauty, Beauty is not love, Love is not music and Music is THE BEST” —Frank Zappa

11 November 2014
How: Methods

Approximate
  Good enough
  Loose specs
  Lazy/speculative

Incremental
  Compose (indirect, virtualize)
  Iterate
  Extend

Divide & conquer
  Abstract with interfaces
  Recursive
  Atomic
  Concurrent
  Replicated

AID
Precise vs. approximate software. Which kind is yours?
Precise: Get it right (avionics, banks, Office)
Approx: Get it soon, make it cool (search, shopping, Twitter)

Features ↔ TTM ↔ speed ↔ cost ↔ dependability ↔ coolness
F^6: Fancy ↔ First ↔ Fast ↔ Frugal ↔ Faithful ↔ Fun
Is it right? ↔ does it run? ↔ will it sell? ↔ can it evolve?

Adaptable: evolving ↔ fixed, monolithic ↔ extensible
Dependable: reliable ↔ flaky; stochastic ↔ deterministic
Coordinate Systems and Notation

Choose the right coordinate system
Like center of mass for dynamics, or eigenvectors for matrices
Example: State as being vs. becoming—(name→value) map vs. log
  — Bitmap/display list; redo-undo log; replicated state machine
Example: Function as code vs. table vs. overlay
  — Table: Cache code results. Overlay: write buffer, search path

Use a good notation
Vocabulary: Types and methods.
Syntax: Domain-specific languages
Primitives: Relations include functions, graphs, tables, state transitions

A point of view is worth 80 points of IQ. —Alan Kay
Science is not there to tell us about the Universe,
but to tell us how to talk about the Universe. —Niels Bohr
Write a Spec

At least, write down the state—Abstract state is *real*

Example: File system state is \( \text{PathName} \rightarrow \text{ByteArray} \)

Then, write down the interface actions (APIs), which ones are external, and what each action \( \pi \) does

Next, write the *abstraction function* \( F \) from code to spec

Finally, show that each action \( \pi \) preserves \( F \):

\[
\text{spec} \quad F(t) \xrightarrow{\pi} F(t')
\]

\[
\text{code} \quad t \xrightarrow{\pi} t'
\]

\[
\text{pre-state} \\ \text{post-state}
\]

*The purpose of abstracting is not to be vague, but to create a new semantic level in which one can be absolutely precise.* —Dijkstra
What: Goals

Simple
Timely (to market)*
Efficient
Adaptable*
Dependable
Yummy*

*More important today
STEDAY: Simple–KISS

Why is it important? Because we can’t do much
Simple is hard, often not rewarded—“That’s obvious.”
Why didn’t computer scientists invent the web?
Why did we invent the Internet?
Simple enough: I can still understand it
But what happens when the system evolves?
Only abstraction and interfaces can save you

How? Interfaces, atomic (D), extensible (I), good enough (A)

Less is more. —Browning
Everything should be as simple as possible, but no simpler. —Einstein
I’m sorry I wrote you such a long letter; I didn’t have time to write a short one. —Pascal
STEADY: Timely—Keep it real

Good enough is good enough

The web is successful because it doesn’t have to work.
Many errors are not fatal
  - They can be retried, automatically (end-to-end) or by the user
  - They can be undone
  - They don’t matter much: Look at Amazon’s web pages

Learn what customers really want—Iterative development

How? Focus (D), extensible, iterate (I), good enough (A)

The best is the enemy of the good. —Voltaire
If you don’t think too good, don’t think too much. —Ted Williams
Perfection must be reached by degrees; she requires the slow hand of time. —Voltaire

And the users exclaimed with a laugh and a taunt,
“IT's just what we asked for but not what we want.” —Anonymous
STEADY: Efficient—Reduce waste

Two aspects: for the implementer, and for the client
   Not unrelated: the client wants it fast and cheap enough

Efficient *enough*, not optimal

Understand what’s important for *you*
   People cost to administer? Standardize, automate.
   Hardware cost to provide a stable service? Write tight code.
   NRE/TTM? Use big components, burn hardware, good enough

**How?** Concurrent (D), shared, deltas (I), lazy (A)

*An efficient program is an exercise in logical brinkmanship.* —Dijkstra
*It’s cheaper to be networked than standalone: continuous updates, shared data, and availability through replication.* —Phil Neches
*I see how it [the phone] works. It rings, and you have to get up.* —Degas
*That, Sir, is the good of counting. It brings everything to a certainty, which before floated in the mind indefinitely.* —Samuel Johnson
Evolution/scaling: Successful systems live a long time
   Machines get faster, load increases, features get added:
   — 2014 PC = 100,000 × Xerox Alto, Web grew from 100 users to 10⁹
Incremental update: Big things change a little at a time
   Databases, web indexes, complex/dynamic displays, routing
Autotuning: Manual is slow, unreliable and expensive
Fault-tolerance: Crashes, errors, bugs are unavoidable

How? Interfaces (D), extensible, distributed (I), loose (A)

Success is never final. —Churchill
One man’s constant is another man’s variable. —Alan Perlis
APL is like a diamond; Lisp is like a ball of mud. —Joel Moses
STEADY: Dependable—Don’t say ‘Sorry’

Reliable: Gives the right answer (safe).
Available: Gives the answer promptly (live).
Secure: Works in spite of bad guys

How much dependability? It depends on the customer
British railways: $1B/life saved
Phone system: much less now than in 1980

Often dependable undo is the most important thing

How? Replicate, partition (D), simple (S), redo log (I)

But who will watch the watchers? She'll just begin with them and buy their silence. —Juvenal
The unavoidable price of reliability is simplicity. —Tony Hoare
How: Methods

Approximate
  Good enough
  Lazy/speculative
  Loose specs

Incremental
  Compose (indirect, virtualize)
  Iterate
  Extend

Divide & conquer
  Abstract with interfaces
  Recursive
  Replicated
  Concurrent
  Atomic

AID
AID: Divide & Conquer

Abstract with interfaces: Divide by difference
Limit complexity, liberate parts. TCP/IP, file system, HTML
Platform/layers. OS, browser, DB. X86, internet. Math library
— Platform as simplifier: Transactions, garbage collection
Declarative. HTML/XML, SQL queries, schemas
— The program you think about takes only a few steps
Synthesize a program from a partial spec. Excel Flashfill
— Signal + Search → Program

Don’t tie the hands of the implementer. —Martin Rinard

Civilization advances by extending the number of important operations which we can perform without thinking about them. Operations of thought are like cavalry charges in a battle — they are strictly limited in number, they require fresh horses, and must only be made at decisive moments. —Whitehead
**AID: Divide & Conquer**

Abstract: Divide by **difference**

Recursive: Divide by **structure**. Part ~ whole
- Quicksort, DHTs, Path names. IPV6, file systems

Replicate: Divide for **redundancy**, in time or space
- Retry: **End to end (TCP)**. Replicated state machines.

Concurrent: Divide for **performance**
- Stripe, stream, or struggle: **BitTorrent, MapReduce**

*If you come to a fork in the road, take it.* —Yogi Berra
*To iterate is human, to recurse divine.* —Peter Deutsch
AID: Incremental

Compose relations, functions, processes, components
   Join, connect, fork
Indirect: Control name $\rightarrow$ value mapping
   - Virtualize/shim: VMs, NAT, USB, app compat, format versions
   - Network: Source route $\rightarrow$ IP addr $\rightarrow$ DNS name $\rightarrow$ service $\rightarrow$ query
   - Symbolic links, register renaming, virtual methods, copy on write

Iterate design, actions, components
   Redo: Log, replicated state machines (state as becoming)
   Undo: File system snapshots, transaction abort
   Scale: Internet, clusters, I/O devices

Extend: HTML, Ethernet

Any problem in computing can be solved by another level of indirection. —David Wheeler
Compatible, adj. Different. —The Devil’s Dictionary of Computing
AID: Approximate

Good enough. Web, search engines, IP packets
   Often non-deterministic
   Eventual consistency. DNS, Dynamo, file/email sync

Loose coupling: Springy flaky parts. Email, Fedwire

Brute force. Overprovision, broadcast, scan
   Reboot: Crash fast
   Strengthen (do more than is needed): Redo log, coarse locks

Relax: small steps converge to desired result.
   Routing protocols, daily builds, exponential backoff

Bottleneck performance analysis—back of the envelope

Hints: Trust, but verify.

Lazy/speculative: bet on future. OCC, write buffer, prefetch

I may be inconsistent. But not all the time.—Anonymous
Summary

Hints and principles—suggest vs. demand

STEADY by AID

What: Simple, Timely, Efficient, Adaptable, Dependable, Yummy
How: Approximate, Incremental, Divide & conquer

If you only remember three things:
  Keep it simple
  Abstract with interfaces
  Write a spec

One last hint: Get it right

If I have seen further than others, it is because I have stood on the shoulders of giants.
—Schoolmen of Chartres, via Newton
The only thing new in the world is the history you don’t know. —Harry Truman
History doesn’t repeat, but it rhymes. —Mark Twain