

Is design important? 75%-80% of system errors are created in the analysis and design phases.

Analysis and design phases account for about only 10% of the overall system cost.

Only about 25% of software projects result in working systems.

(Perhaps you get what you pay for.)

WWMCCS network (Nov 9, 1979, potential global thermonuclear war)^[1]
(human error loaded training tape into active system)

Therac-25 Medical Linear Accelerator (1985-1987, ≥ 3 deaths)^[2]
(software safety controls, hardware interlocks, testing failures)

Patriot Anti-missile Timing Bug (1991, 28 deaths, 98 wounded)^[3]
(concurrency bugs, numeric precision limitations)

You have a very detailed specification for the GIS system.

That's actually very different from a typical experience in a the workplace.

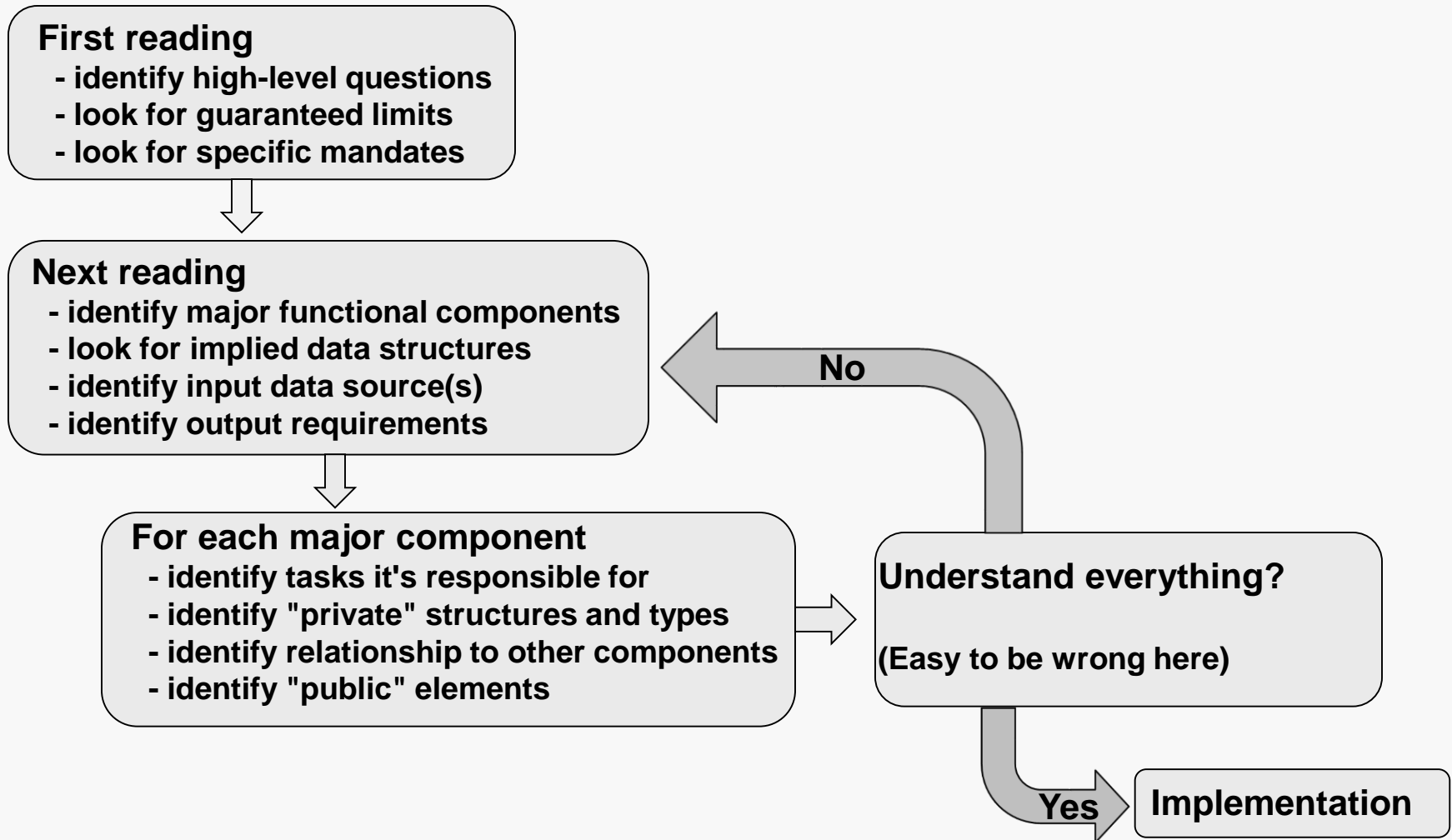
There, you're likely to have a client with:

- some vague notions of what is actually needed (or desired)
- little ability to clearly express those vague notions
- no understanding of the limitations imposed by hardware and language
- solid protections against the more effective interrogation techniques

Here, you have:

- lots of details regarding what is required
- test data and reference output to use

I'd say the right way to approach the specification is something like this:



The rest of this presentation shows ONE approach to analyzing a particular specification.

It is certainly not the only way.

It is also not a complete rendition of the design I used, but it does imply the rest.

The discussion leaves some useful questions unidentified, and therefore unasked.

It does not discuss coding issues at all... they generally have absolutely nothing to do with the creation of a design.

Here are a few that are easily found in the specification:

Must be able to deal with a specified set of search queries.

Must achieve a certain efficiency in dealing with queries.

Must use arrays for data structures.

Never need to deal with more than a specified number of data records.

Length limits on certain record fields.

It helps to visualize the system and imagine it in operation.

dB file (file of GIS records)

script file (file of commands to be serviced)

log file (file holding results of command servicing)

command processor (uses other components to execute commands)

feature name index (map feature name + state abbreviation to dB file offset)

feature ID index (map feature ID to dB file offset)

Sometimes the analysis of a components reveals the need (or benefit) of including more subtle components:

command processor (uses other components to execute commands)

index (provides interface for index lookups, encapsulates lower-level indices)

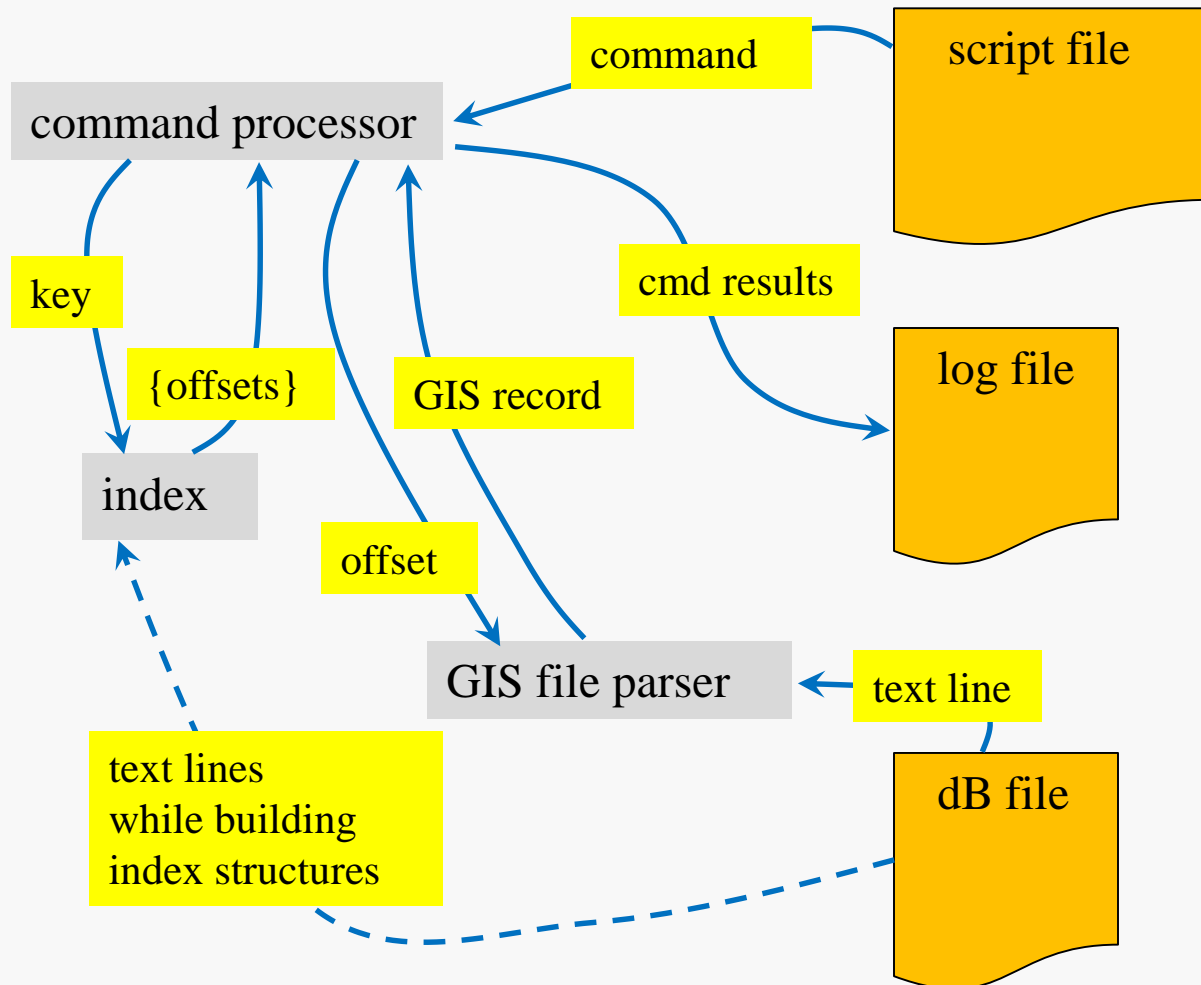
feature name index (map feature name + state abbreviation to dB file offset)

feature ID index (map feature ID to dB file offset)

GIS file parser (front end for accesses to the dB file)

- isolates handling formatting of GIS records

We need to understand which components interact, and what communication takes place.



Now, here's how I envision the system at work:

Phase 0:

Verify input files exist

Phase 1:

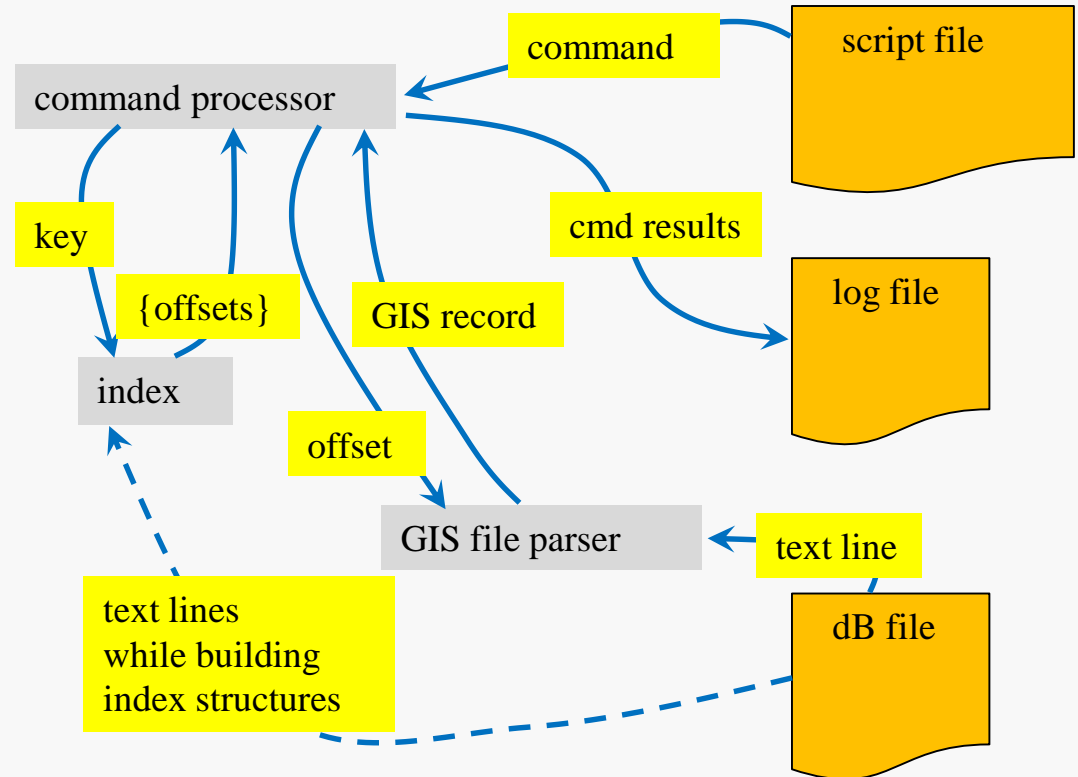
Build Name index and
Feature ID index

Phase 2:

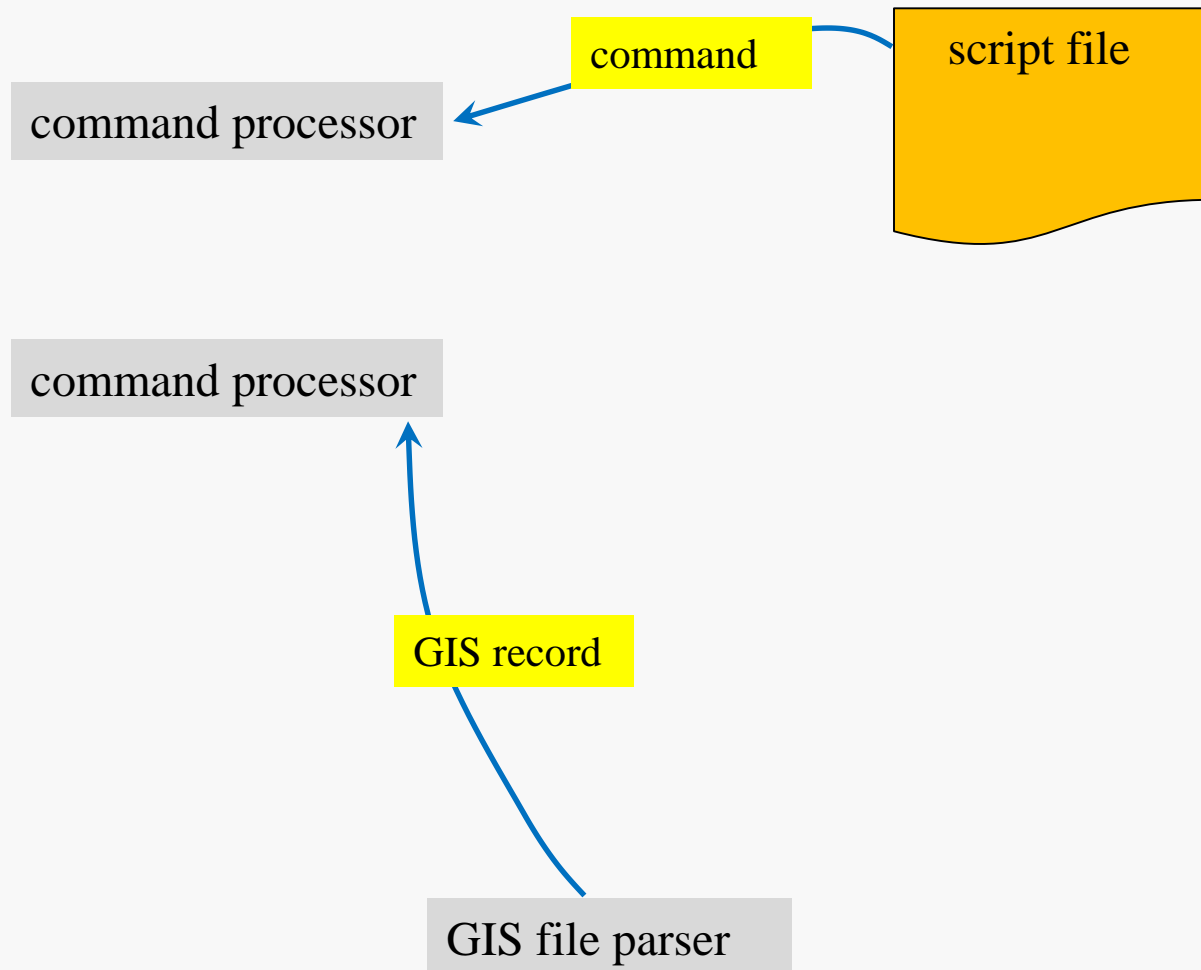
Process commands from
script file

Phase 3:

Close files, deallocate
memory, shut down



Sometimes the analysis of relationships reveals the possible benefit of additional user-defined data types:



A component may require internal types, that would not be meaningful to other components:

feature name index (map feature name + state abbreviation to dB file offset)

array of objects holding:

- a key value
- set of record offsets

index entry object:

- key: depends on feature name and state abbrev?
- how to store a set of record offsets?

A component may require internal parts, that should not be accessible to other components:

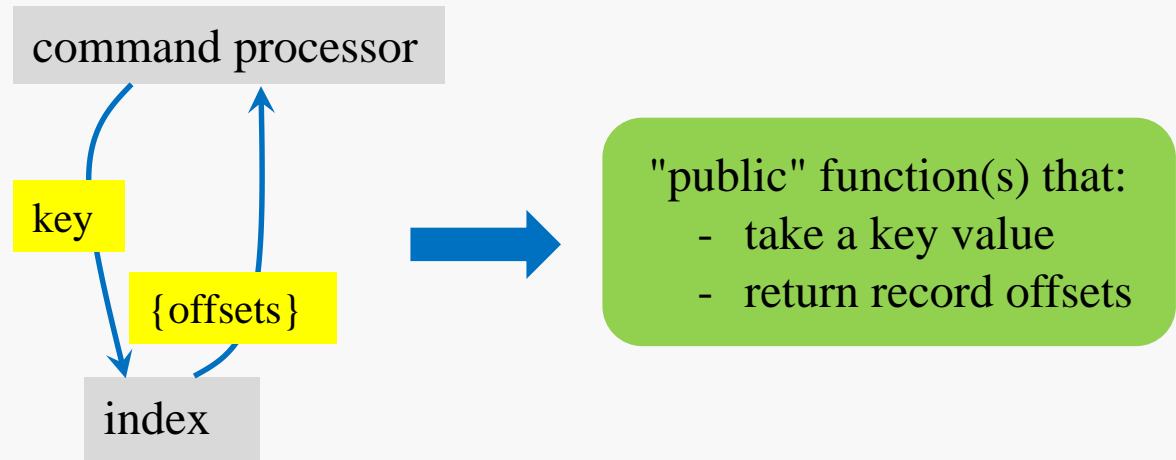
index (provides interface for index lookups, encapsulates lower-level indices)

feature name index (map feature name + state abbreviation to dB file offset)

feature ID index (map feature ID to dB file offset)

These should NOT be accessed directly by other components.

Each interaction requires at least one "public" function to support it:



A *module* is essentially a way of packaging the types, structures, and functions needed to provide a certain aspect of functionality.

Generally, I would consider each component identified earlier to be a potential module, leading to a C source file and an accompanying C header file.

Probably, no design is perfect.

Probably, every design could be improved.

By adding something overlooked earlier.

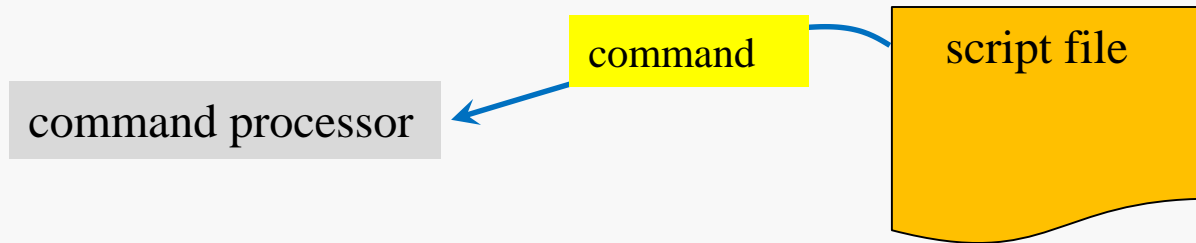
By removing something identified earlier..

But, eventually we need to move on to creating an implementation.

Frequently you will discover design flaws during the implementation phase.

That can be painful...

... which is just another reason careful attention to design pays off.



Does a Command type make sense?

Does having a Command type make anything simpler or more efficient?

What entity would create a command object?

Should there be another component in the system design to do this?

All good questions... something to consider...

command processor

GIS record

GIS file parser

Does a GIS record type make sense?

Does a GIS record type make anything simpler or more efficient?

How am I going to acquire the different GIS record fields?

```
1481851|Blowing Springs Campground|Locale|VA|51|Bath|017|380408N|0795304W|. . .
```

One option:

Grab the whole line as a string: `fgets()`

Parse the string into fields: `sscanf()` or `strtok()` or ??

Another option:

Grab the fields one by one from the file: `fscanf()` or ??

How am I going to deal with file offsets?

Some useful functions:

```
ftell()
```

```
fseek()
```

Look at what's available in `stdio.h` (pubs.opengroup.org)

- [1] *Close Calls with Nuclear Weapons*, Union of Concerned Scientists

www.ucsusa.org/sites/default/files/attach/2015/04/Close%20Calls%20with%20Nuclear%20Weapons.pdf

Command and Control: Nuclear Weapons, the Damascus Accident, and the Illusion of Safety, Eric Schlosser, Penguin Books, 2014 (978-0143125785)

- [2] *An Investigation of the Therac-25 Accidents*, Nancy G Levenson, University of Washington and Clark S Turner, University of California, Irvine.

ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=274940

- [3] *Patriot Missile Defense: Software Problem Led to System Failure at Dharan, Saudi Arabia*, IMTEC-92-26

www.gao.gov/products/IMTEC-92-26