Command Graphs

April 3, 2014

Mark A. Overton

Software Engineer

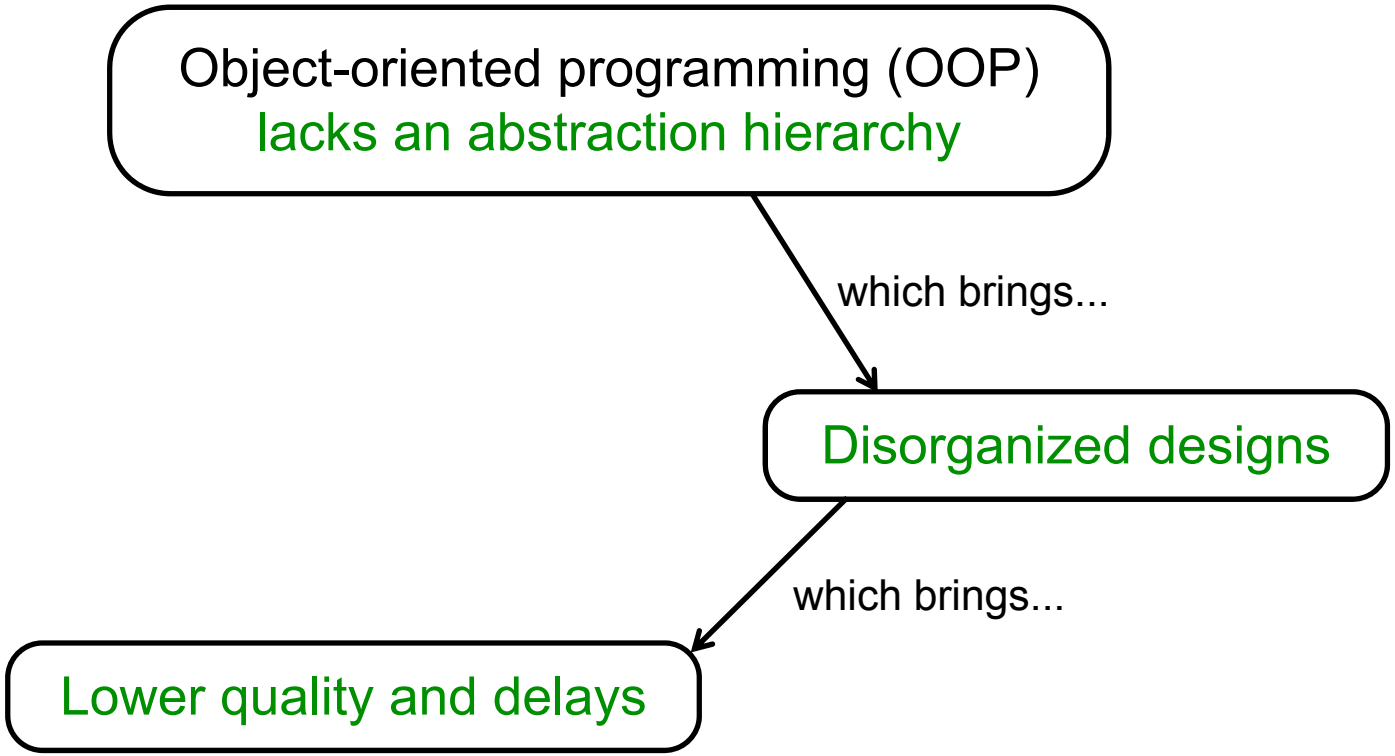
Acknowledgements

Thanks to Jim Ray
and Dorothy Kennedy
for their unswerving support of this effort.

A New Design-Method? Why?

NORTHROP GRUMMAN

Software is often late and buggy.
Here's part of the reason...

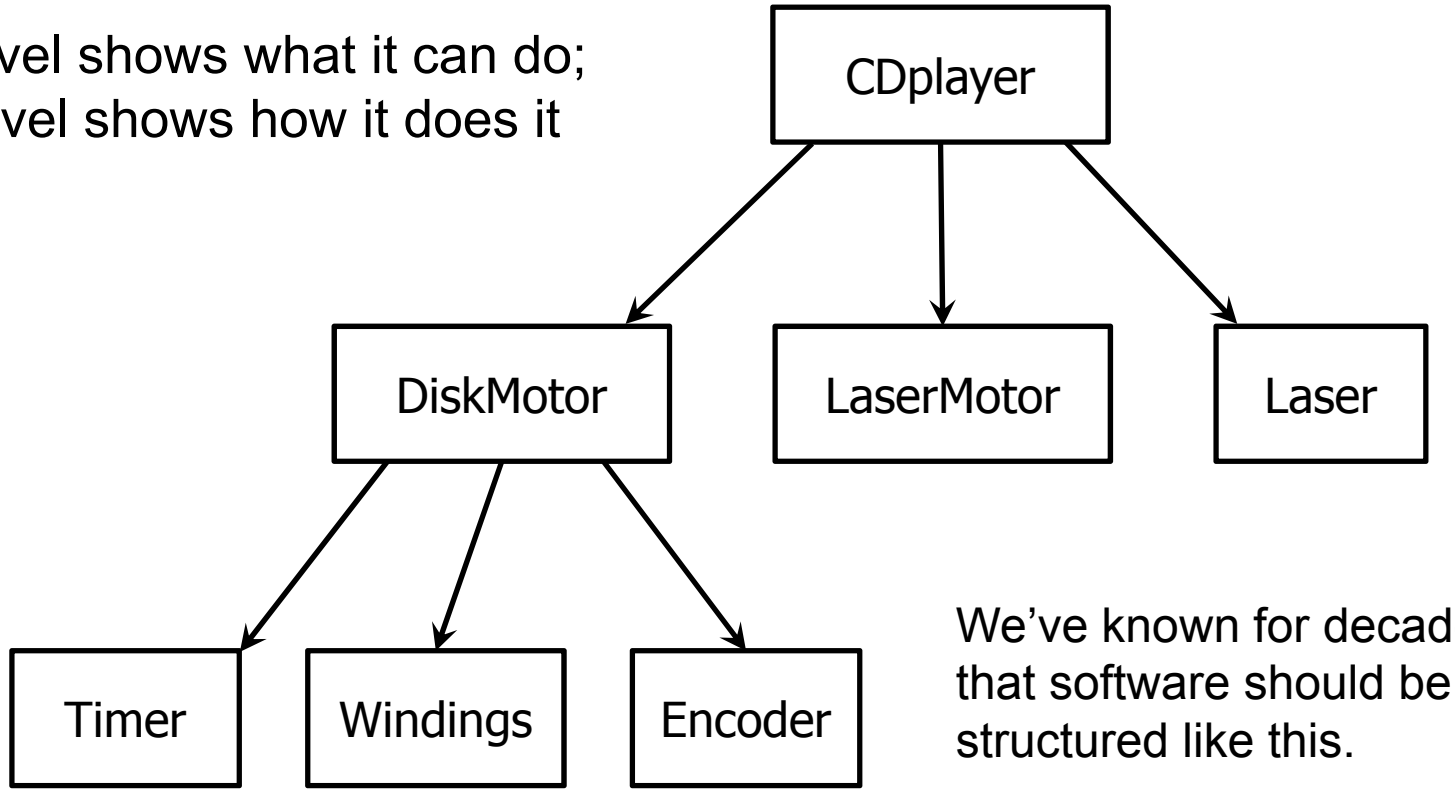


Abstraction Hierarchy is Levels

NORTHROP GRUMMAN

Levels of generality. Levels of capability. Levels of control.

Each level shows what it can do;
lower level shows how it does it



We've known for decades that software should be structured like this.

The Chasm of Disorder in OOP

NORTHROP GRUMMAN

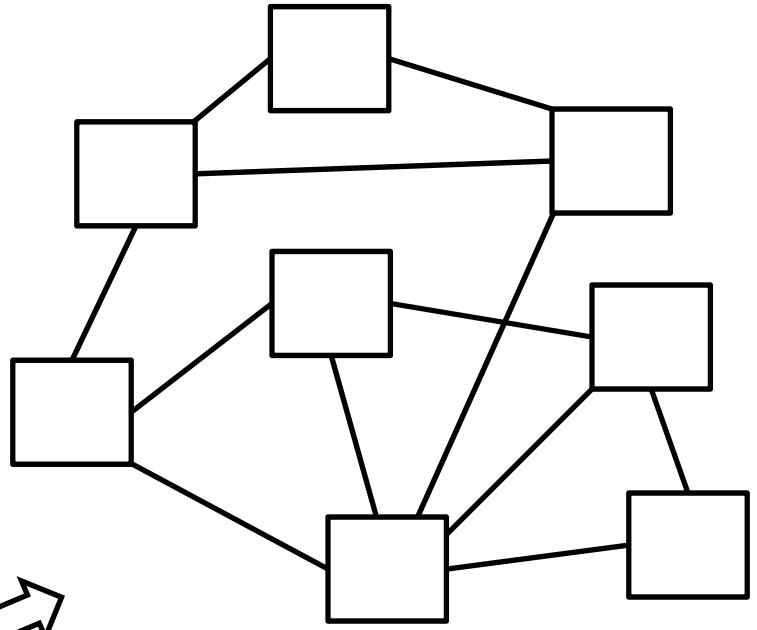
highest level

system

subsystem (not hierarchical in UML)

chasm of disorder

In this chasm, designers are unguided and unconstrained, so they aimlessly create and connect objects, producing disorder like this spaghetti:



class and object

subroutine/method

lowest level

The chasm encourages disorder

Debugging the Mess

NORTHROP GRUMMAN

No abstraction hierarchy ⇨ Disorder ⇨ Most time is spent debugging the mess

Result is buggy and late software

Additional notes:

Messy designs are hard to understand, which tends to make them even messier.

I have ported much OO software. Most has been messy because its designers were not guided into designing with levels within the chasm. Such designs look like spaghetti.

The Solution

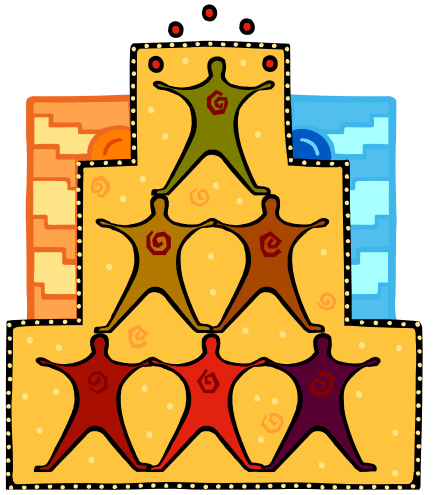
NORTHROP GRUMMAN

Enforce a hierarchy of abstractions within the chasm

- But don't excessively limit flexibility
- Created a new way to design OO software based on I-D-A-R rules:

  Identify – Down – Aid – Role

- Don't be repelled by more rules; these are instinctive and easy to remember



- Two fielded programs conform to these rules.
  10 life-sized designs have been created. This method actually works!

Rule #1: Identify

NORTHROP GRUMMAN

Identify all methods in classes as commands or notices

- Commands are imperative: They initiate action. Each starts with a verb.
  Examples: *playCD, sendMsg, awaitButton*

- Notices are informative: They provide info such as events, status, ...
  Examples: *atSpeed, msgArrived, buttonWasPressed, sensorData*

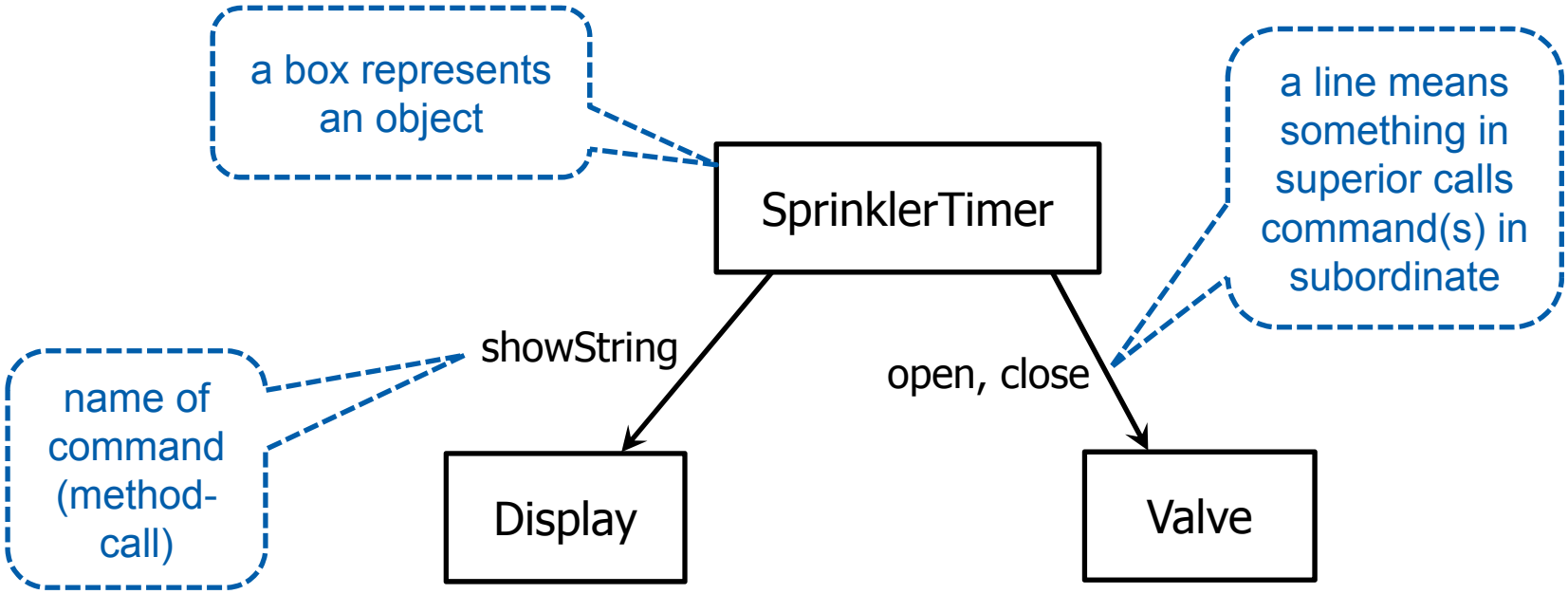
Additional notes:
Constructors/destructors and private methods are exempt. Non-call communications such as mailbox-messages are also ID'd as commands or notices.
It appears that nobody ever thought to split public methods like this.
"Method" means "subroutine" in OOP-lingo.

Rule #2: Down

NORTHROP GRUMMAN

Draw calls to commands pointing down

Produces a hierarchy of commands, like a command-hierarchy in the military.

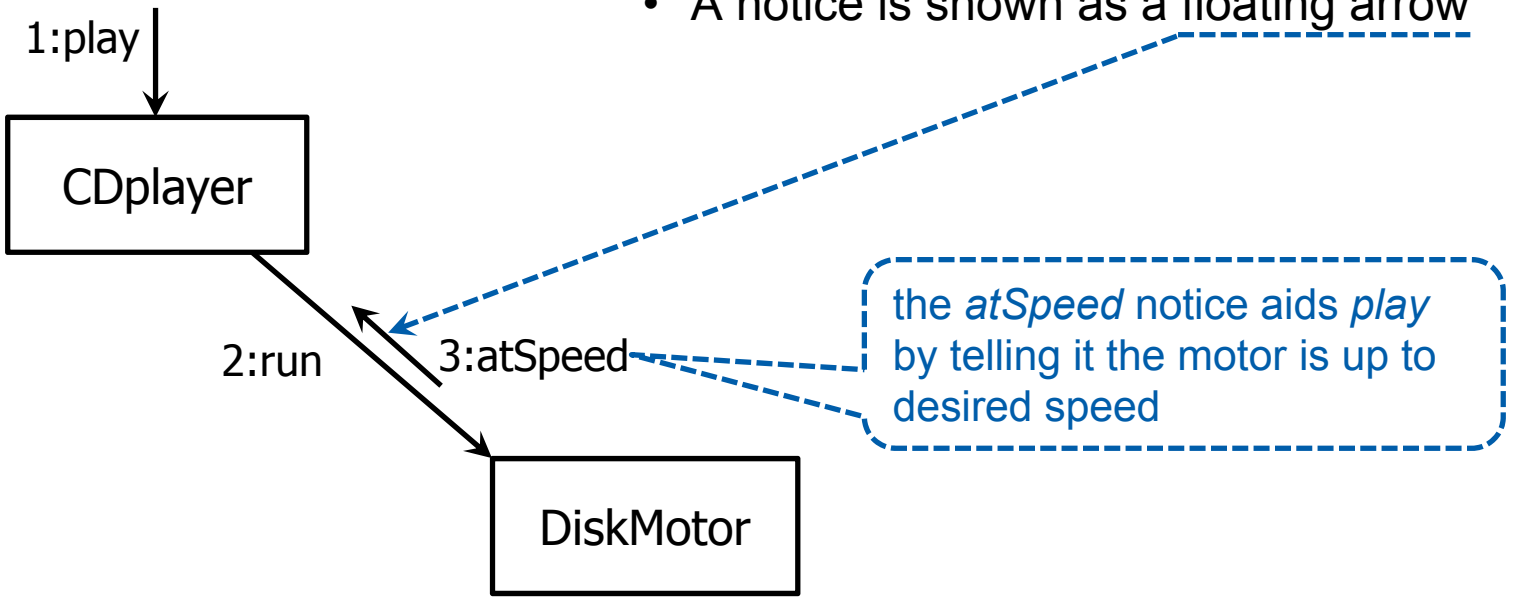


Rule #3: Aid

NORTHROP GRUMMAN

A notice may only aid command(s) in its class

- A notice always aids a command by providing information it needs (need-to-know)
- A notice may also perform actions on behalf of a previously-called command
- A notice is shown as a floating arrow



Rule #4: Role

NORTHROP GRUMMAN

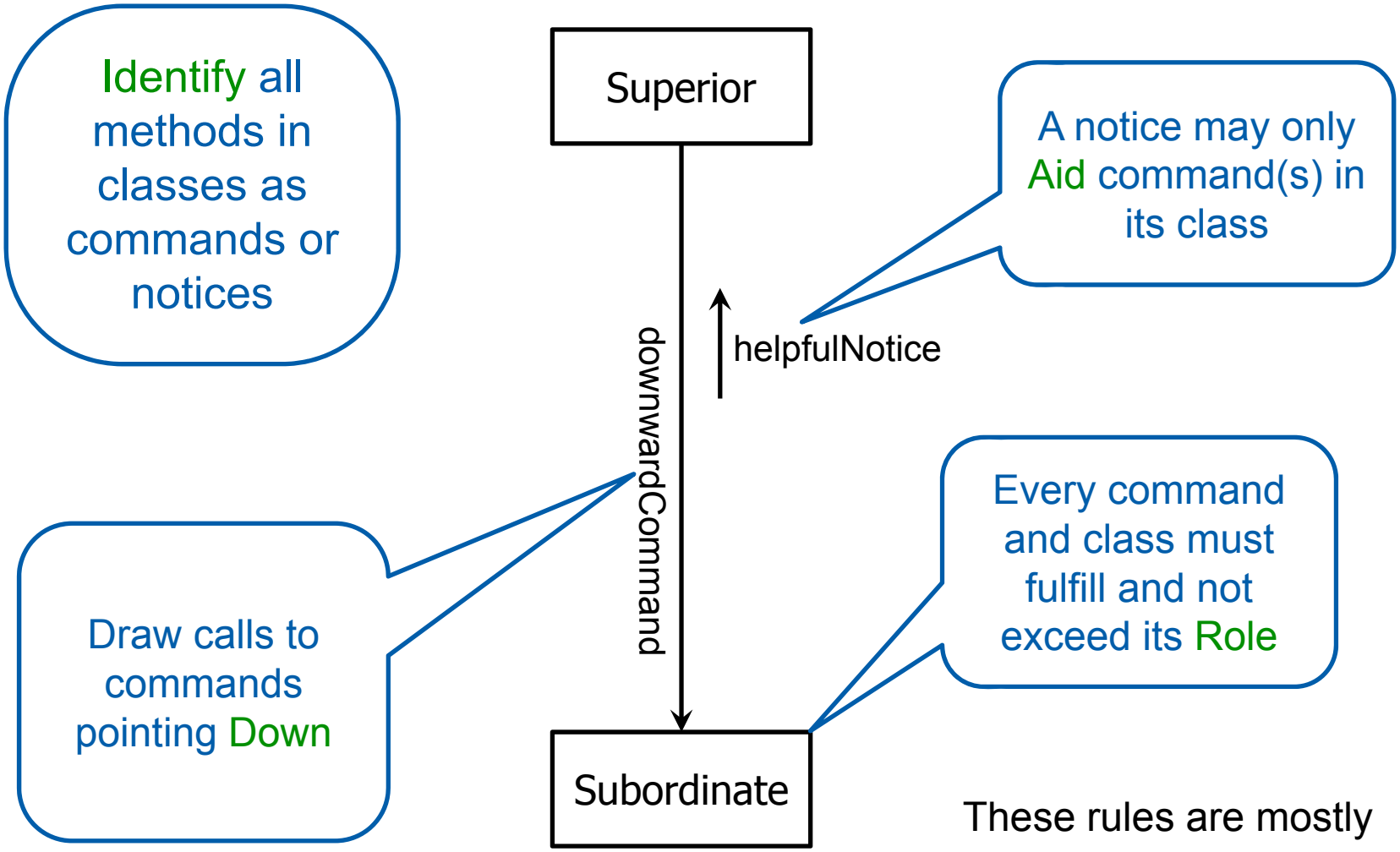
Definition:
A “role” describes the gist (summary) of what a command or class does.

Every command and class must fulfill and not exceed its role

- Example: Role of DiskMotor class is “spins motor at a given speed”
- This rule causes a command or class to implement an abstraction instead of being a hodge-podge of poorly-related actions
- Cross-cutting concerns are an exception (debug, logging, ...)

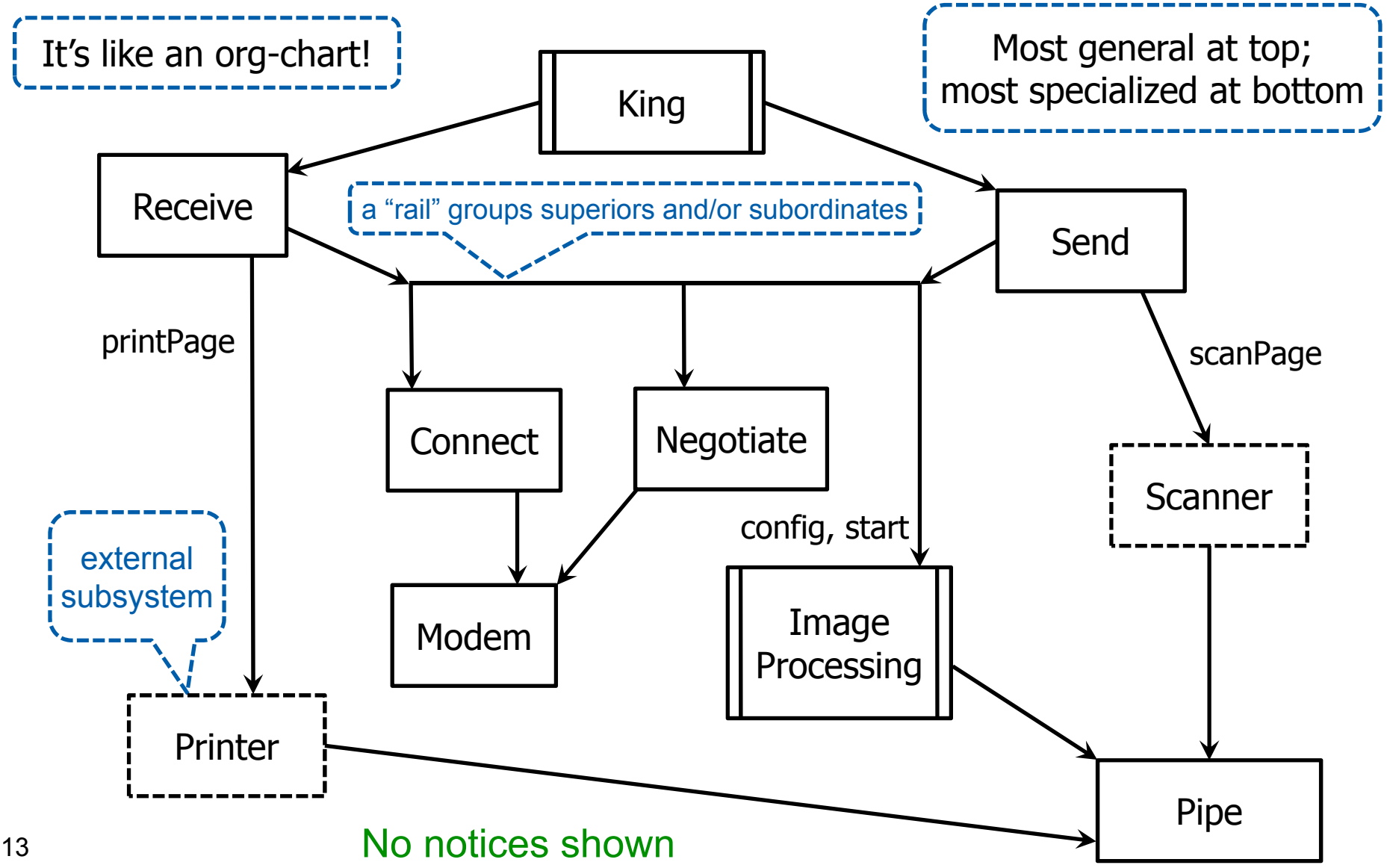
The I-D-A-R Rules are Easy

NORTHROP GRUMMAN



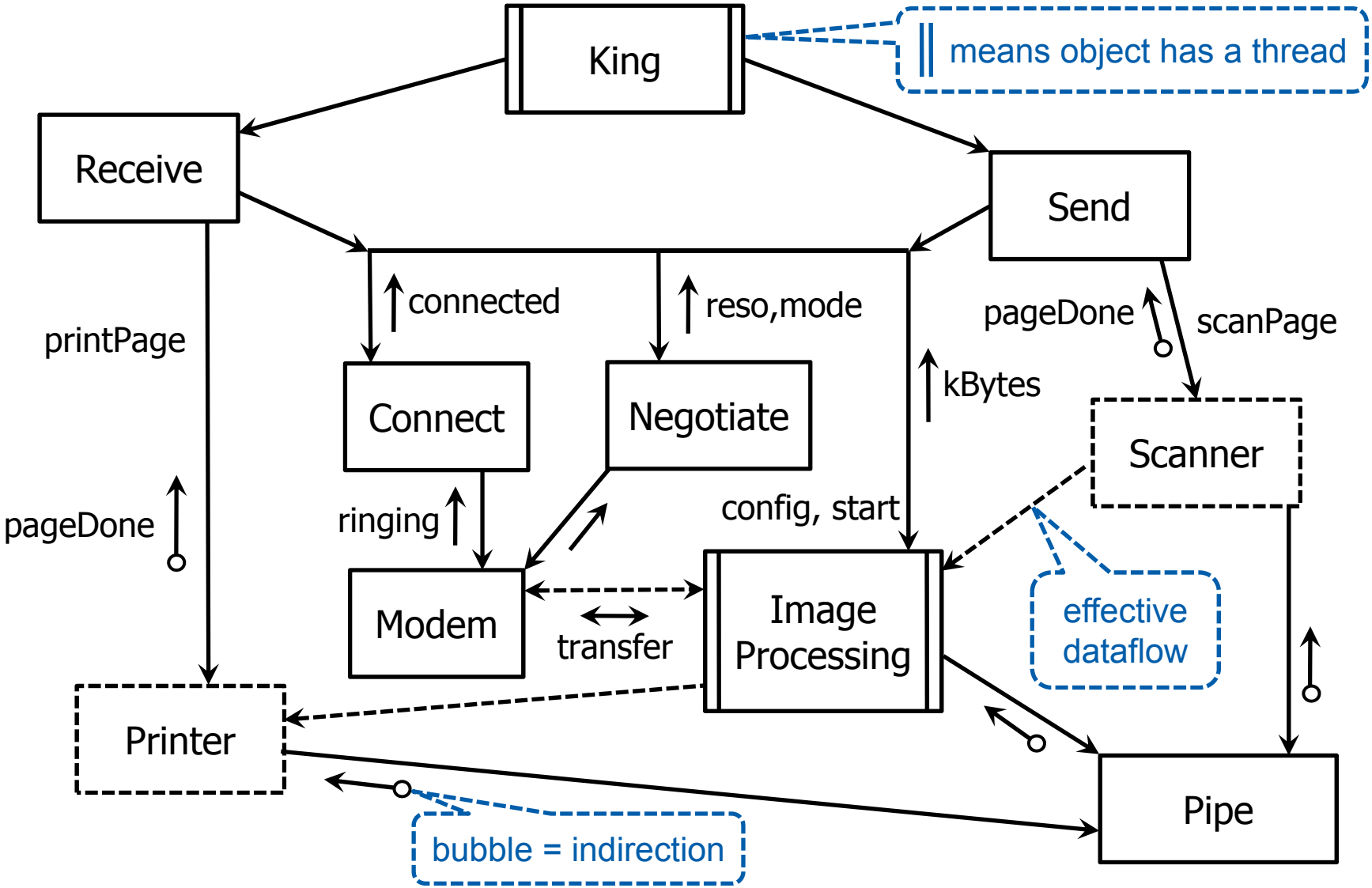
These rules are mostly common sense

Example Design: Fax Machine



Example Design: Fax Machine

NORTHROP GRUMMAN

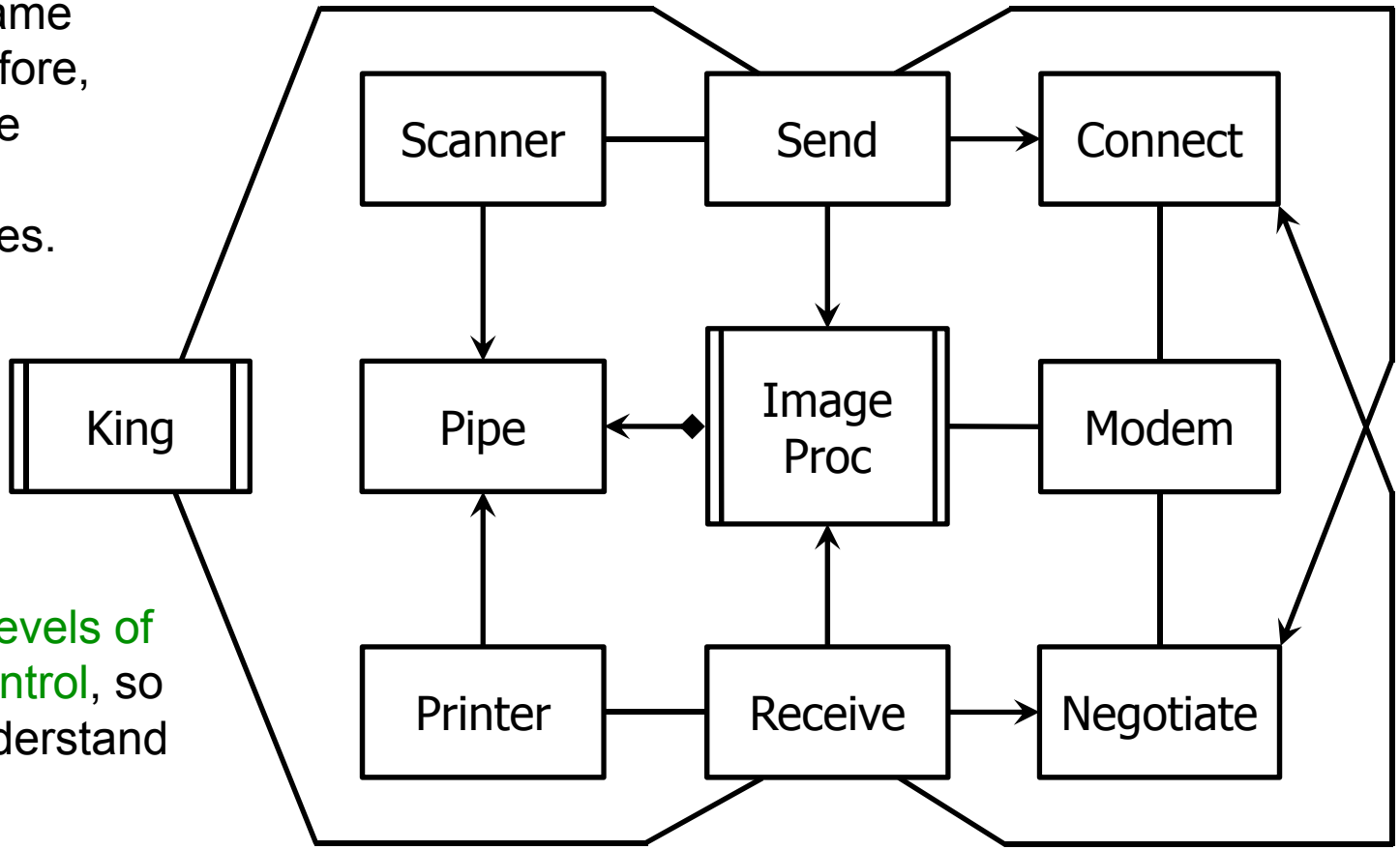


UML Diagram of Fax Machine

NORTHROP GRUMMAN

UML is prettier, but less helpful. (UML is the standard way of diagramming object-oriented software)

This is the same design as before, with the same connections among classes.

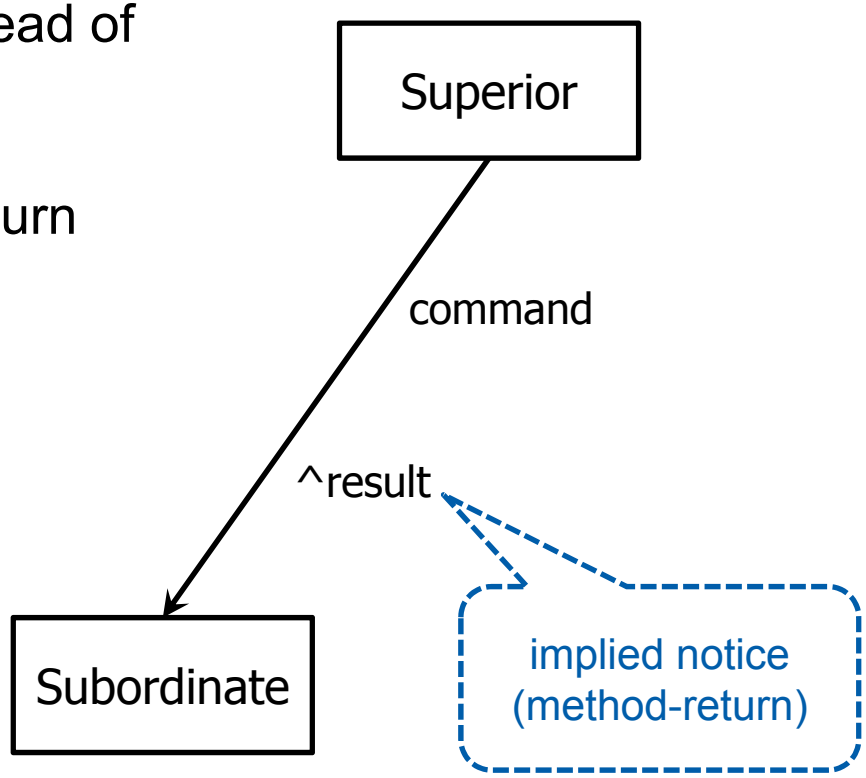


It shows no levels of generality/control, so you don't understand how it works.

Implied Notices

NORTHROP GRUMMAN

- Method-return is an implied notice saying “I’m done, and here are the results”
- A caret (^) on label is drawn instead of a floating arrow
- Notice is piggybacking on the return
- Use these as much as possible because they’re simple and minimize dependencies

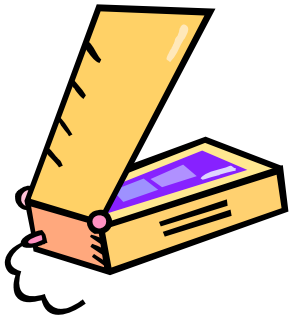


Threads

NORTHROP GRUMMAN

- Commands and notices can be called from any thread, including ISRs
- Use double-lines || for an object containing a thread (same as UML)
- Use a lightning bolt ↯ for an object containing an ISR (interrupt service routine)
- The double-line and lightning bolt notations are enough to tell a reader what threads are used for commands and notices

Example Design: Scanner Subsystem



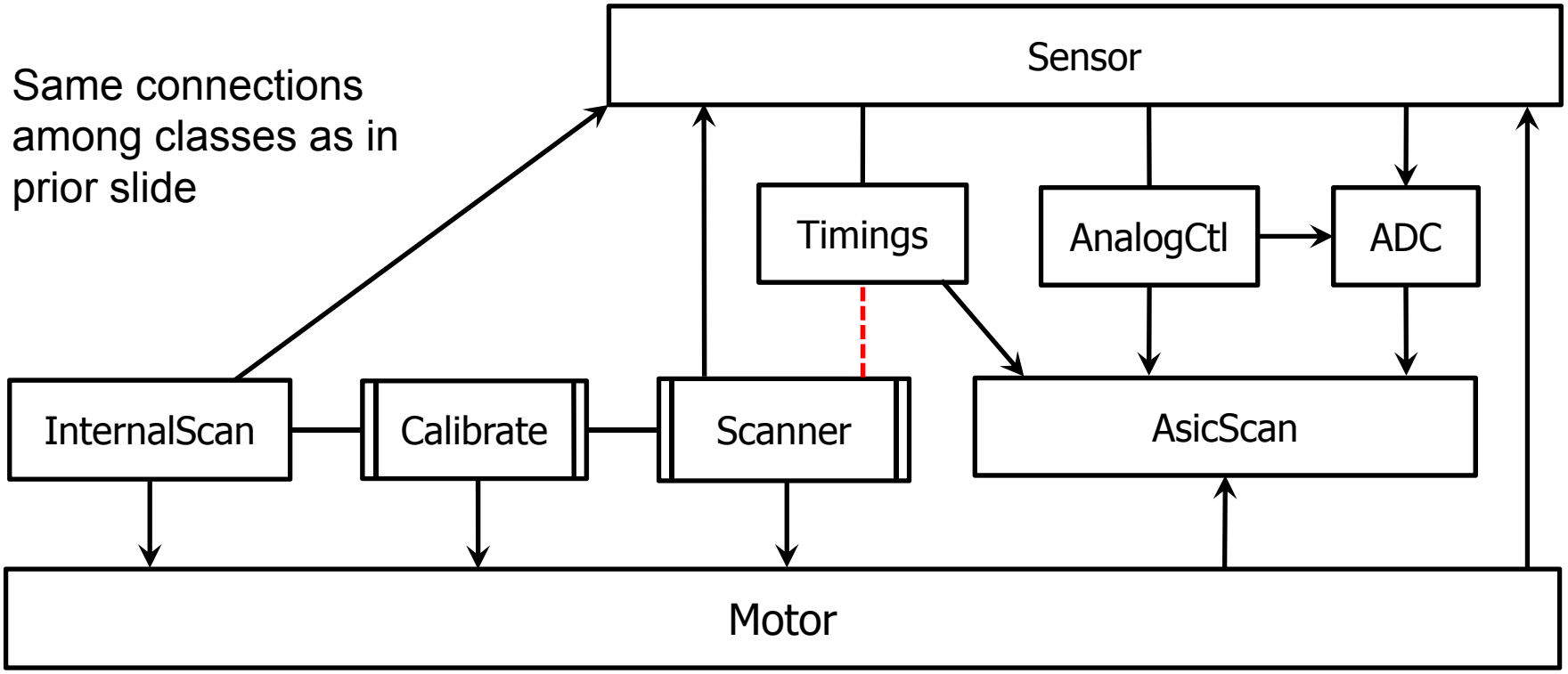
What do you think of that atStartPos notice?

UML Diagram of Scanner Subsystem

NORTHROP GRUMMAN

UML cannot show the most important part of a design – its hierarchy of control.
It's because OOP has no concept of an object controlling another object.

Same connections among classes as in prior slide

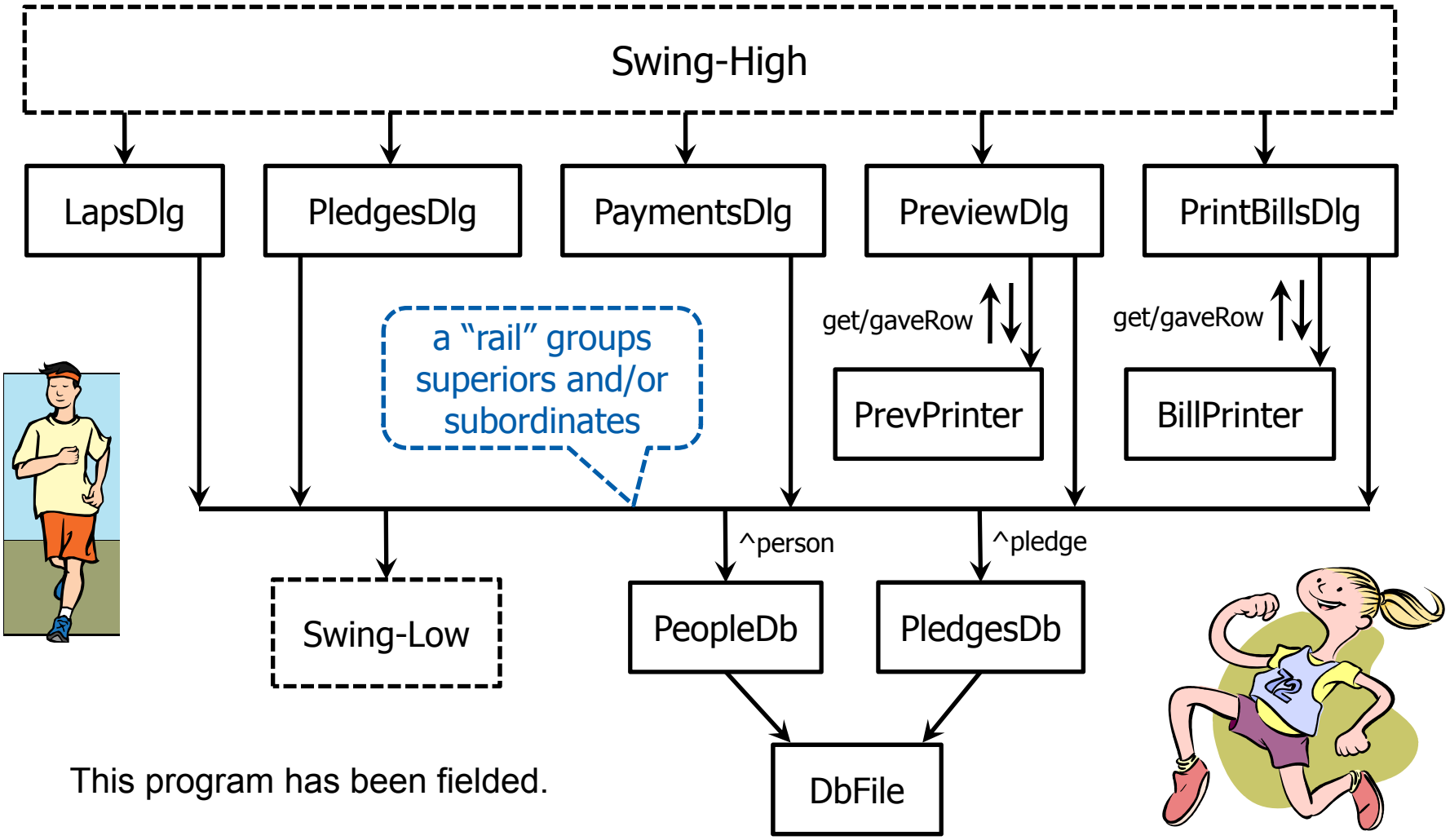


"Control" is out of control: OOP's ignorance of control encourages messy control-structure, because it's OK for anything to control anything.

Example Design: Jog-a-Thon (Java/Swing)

NORTHROP GRUMMAN

The IDAR method works well with GUI-based applications.



This program has been fielded.

These Graphs Show the IDAR Method...

NORTHROP GRUMMAN

- Forces designs to use levels of abstraction, preventing disorder
- Is understandable. They are embellished organization-charts, helping keep designs clean
- Can represent complex real-life designs
- Is not over-constraining or stifling. The four rules are intuitive

Also...

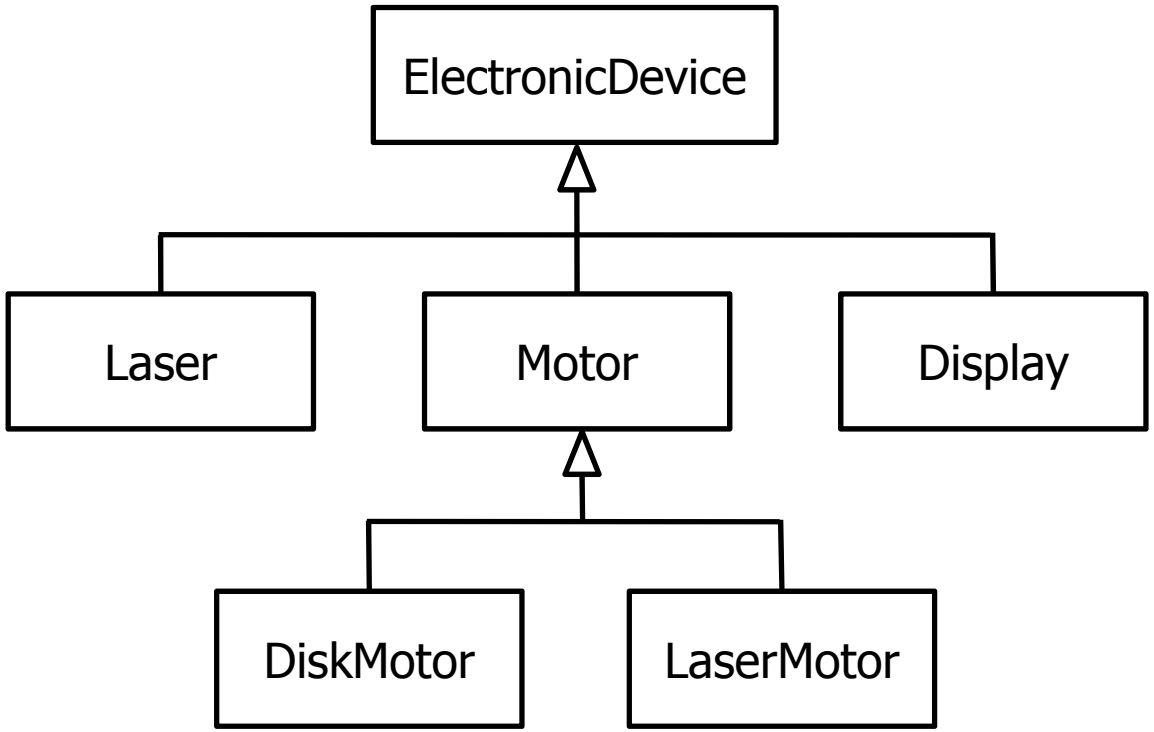
- IDAR graphs can represent various threading schemes

- UML diagrams are much harder to understand (because today's OOP has no hierarchy), and that tends to make today's designs messier

Why not use an Inheritance-Hierarchy?

NORTHROP GRUMMAN

- Inheritance provides a hierarchy of categories or versions
- This is a poor kind of hierarchy,
  because it cannot show what controls what

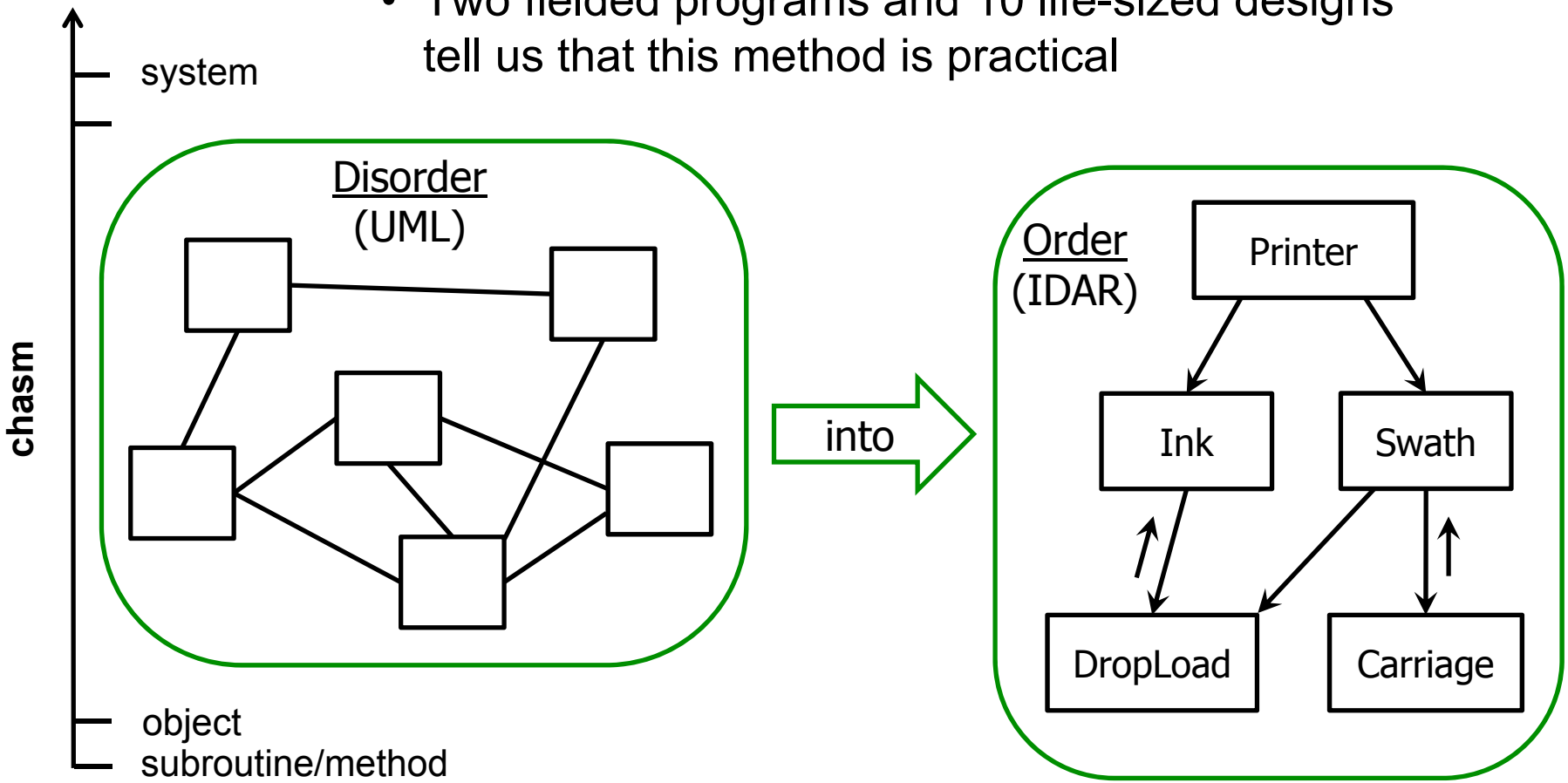


Inheritance has been emphasized because OOP has had no other kind of hierarchy, until now.

Summary

NORTHROP GRUMMAN

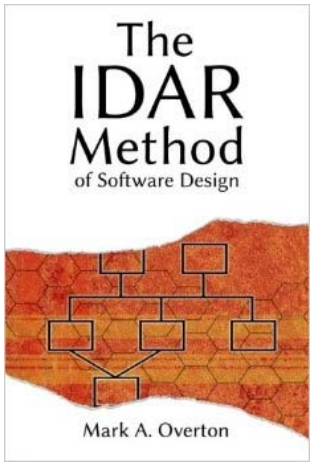
- IDAR method changes disorder into order in the chasm, shortening schedule and improving quality
- Two fielded programs and 10 life-sized designs tell us that this method is practical

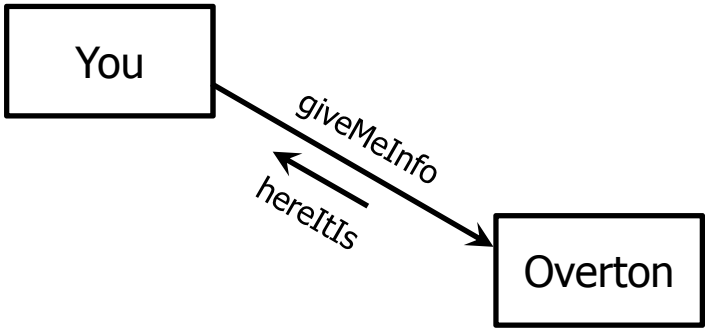


For More Information

NORTHROP GRUMMAN

- Email me: mark.overton@ngc.com
- Visit my website: http://idarmethod.com
  (you might need to type the http://)
- I self-published a 370-page book.
  Search for “The IDAR Method” in amazon.com
- Look at the following slides in this presentation





Appendix (covered if time permits)

NORTHROP GRUMMAN

- New design patterns based on hierarchy
- Similarity to human organizations
- IDAR enables better peer-reviews

Identify	*identify* all methods in classes as commands or notices
Down	draw calls to commands pointing *down*
Aid	notices only *aid* commands (giving them info; optional execution)
Role	every command and class must exactly fulfill its *role*

Design Patterns

NORTHROP GRUMMAN

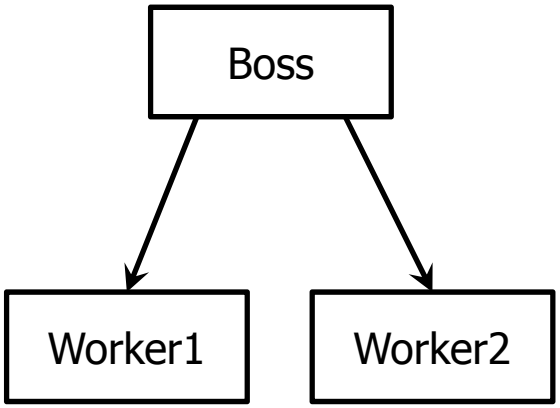
The IDAR method of design unveils some new design patterns that are based on hierarchy.

- Boss – commands workers
- Resourceful Boss – also manages resources
- Watcher – reports arriving messages/events to superior
- Secretary – gives incoming chores to boss when appropriate
- Various dataflows – horizontal/vertical, push/pull
- Dispatcher – routes commands to subordinates w/ same interface
- 20 patterns are described in the book “The IDAR Method”

Boss Pattern

NORTHROP GRUMMAN

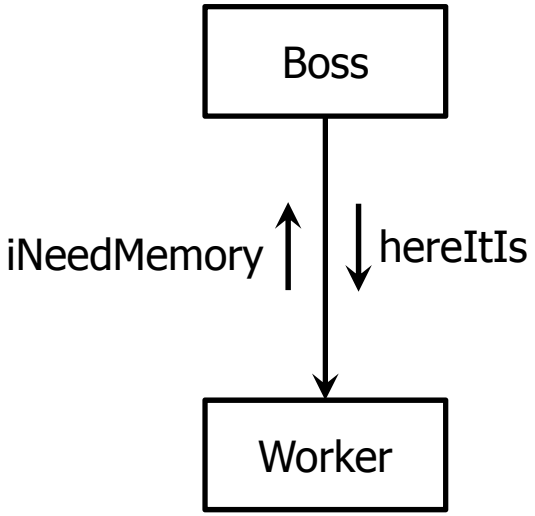
- In a sense, every command follows this pattern
- Boss delegates work to more specialized subordinates



Resourceful Boss Pattern

NORTHROP GRUMMAN

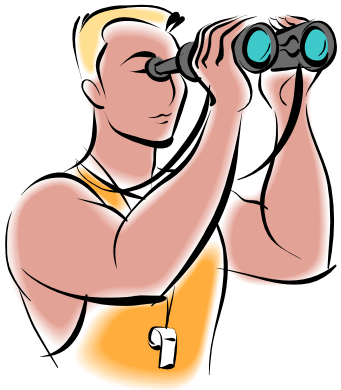
- Same as Boss pattern, but also is responsible for all resources needed by subordinates
- Subordinates request resource via a notice, and boss grants it via a notice
- Eliminates deadlocks and resource-leaks because alloc/free are done in one place

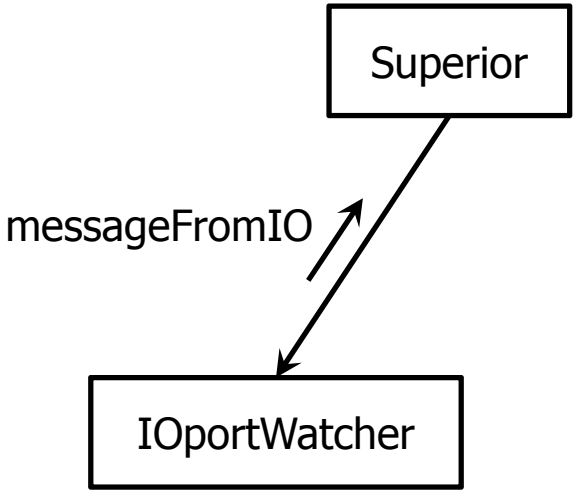


Watcher Pattern

NORTHROP GRUMMAN

- Constantly watches a source of messages or events
- Reports each to its superior as a notice
- Messages may contain commands, giving the appearance that watcher is commanding its superior, but watcher is merely a mail-carrier

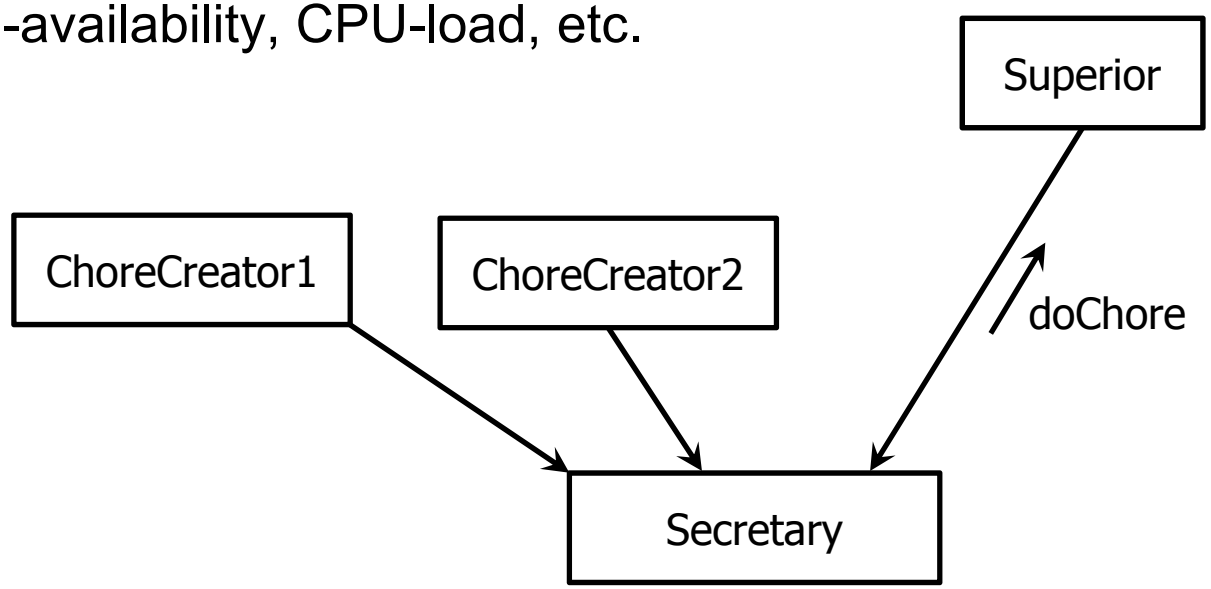




Secretary Pattern

NORTHROP GRUMMAN

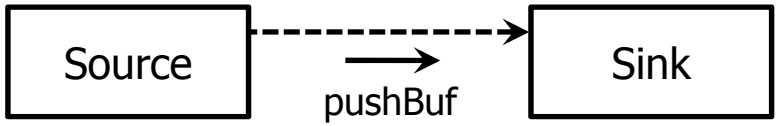
- Stores arriving chores for its superior
- When a chore is ready to be performed, secretary gives chore to its superior in a "do chore" notice
- Chore-readiness can be based on schedule, resource-availability, CPU-load, etc.



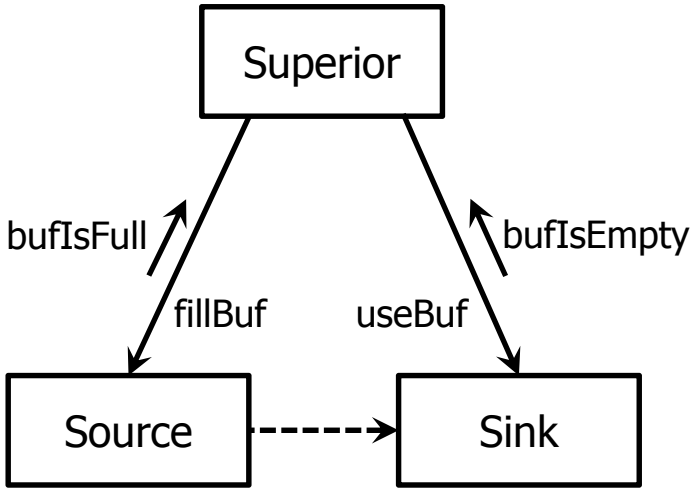
Dataflow Patterns

NORTHROP GRUMMAN

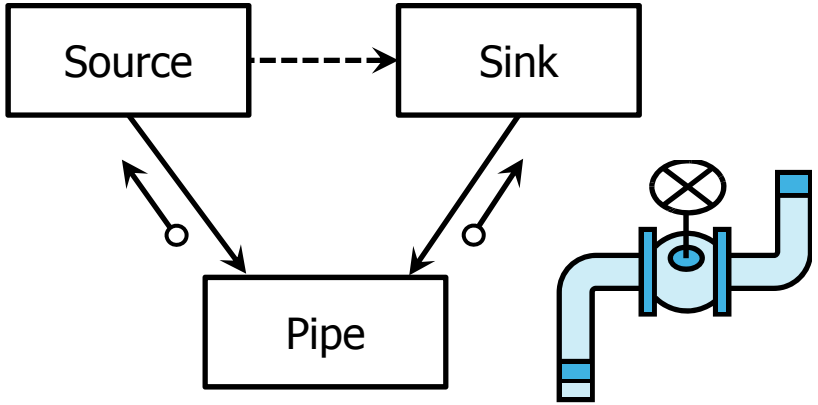
In IDAR graph (command graph), dataflow is shown as dashed arrow



peer-to-peer



through mutual superior

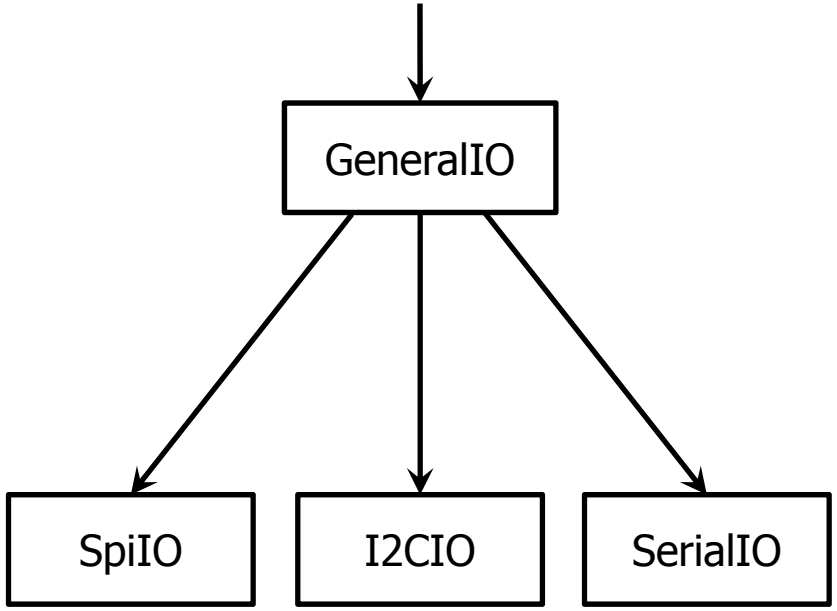


through mutual subordinate (pipe)

Dispatcher Pattern

NORTHROP GRUMMAN

- Routes commands to subordinates having identical interface
- Methods only contain **switch** or **case** statements
- Used to avoid indirect calls (including inheritance), replacing indirect calls with direct calls
- Why avoid indirection?
  Direct calls are easier to follow when reading the source-code, reducing maintenance-cost.

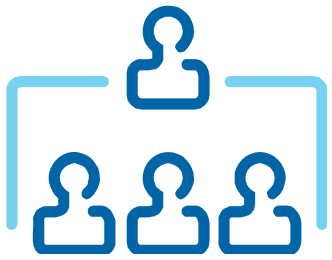


Similar to Human Organizations

NORTHROP GRUMMAN

IDAR Rule	Organizational Rule
Every public method is identified as being a command or a notice	Communications among people are commands or informational
Commands must point down in the graph	People may only command their subordinates
A notice must convey information to aid command(s)	People should give information only to those who need it
Commands must exactly fulfill their roles	People must fulfill their roles in the organization

These parallels with organizations such as armies and corporations indicate that we got it right.



Peer-Reviews Fail (to simplify messes)

NORTHROP GRUMMAN

Managers and leads should review designs!

- Peer reviews fail to simplify designs because peers avoid angering folks they work with. So they only point out bugs and rule-violations, but not excess complexity
- Anyone can understand IDAR graphs (unlike UML).
  Think of them as an org-chart of specialized workers
- Anyone can review them to make sure designs are sensible and as simple as possible
- So managers and leads need to find the excess complexity

Acronyms

NORTHROP GRUMMAN

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
BUF	Buffer
CPU	Central Processing Unit
DB	Database
DLG	Dialog
IDAR	Identify-Down-Aid-Role
IO	Input/Output
ISR	Interrupt Service Routine
OOP	Object-Oriented Programming
UML	Unified Modeling Language