Introduction

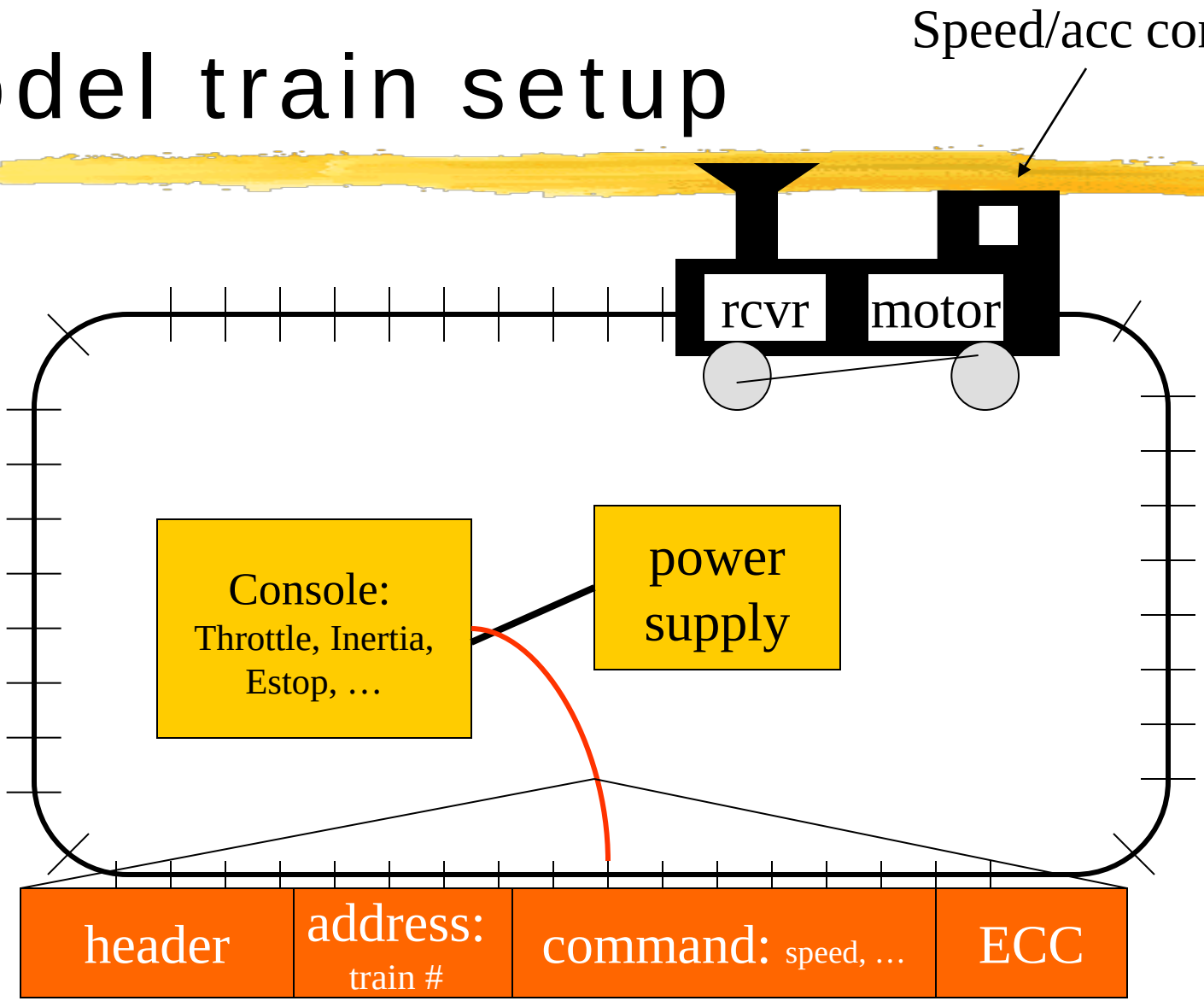
- Example: model train controller.

Purposes of example

- Follow a design through several levels of abstraction.

- Similar design scheme to be used in the course project

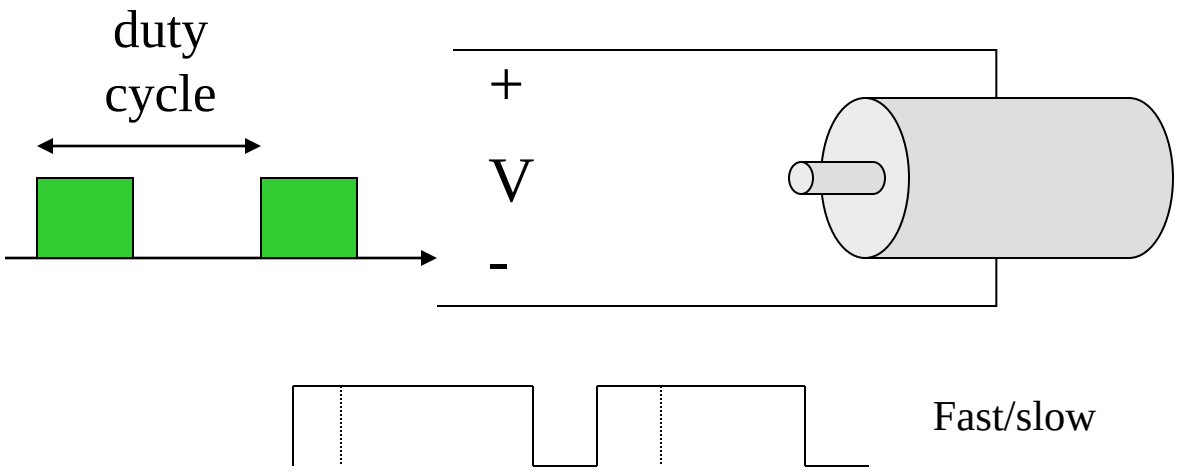
Model train setup



Taken/modified from "Computers as Components, Wolf"

Train speed control

- Motor controlled by pulse width modulation:
- Sketching out the spec first helps us understand the basic relationships in the system.



Taken/modified from "Computers as Components, Wolf"

Requirements

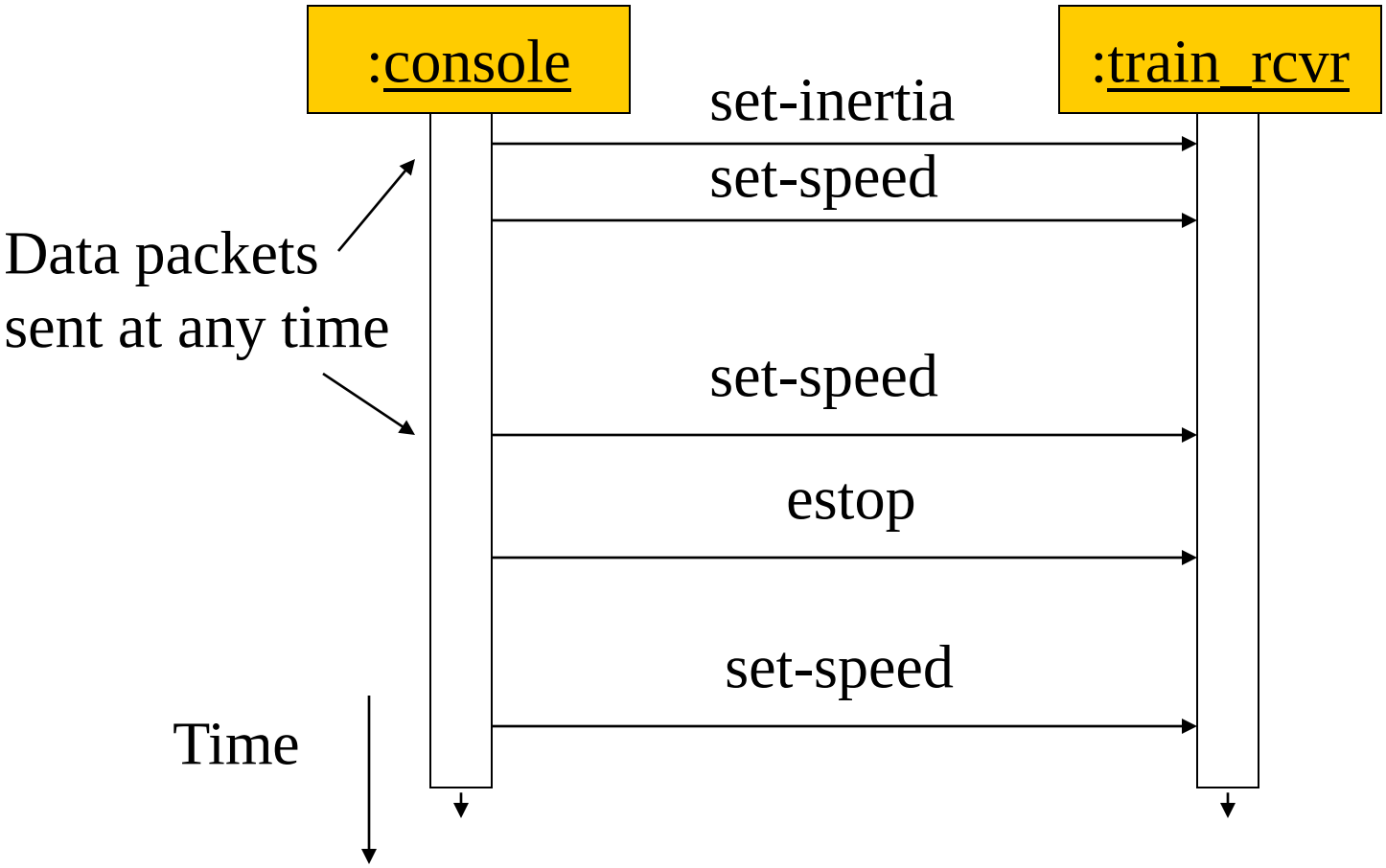
- Console can control 8 trains on 1 track.
- Throttle has at least 63 levels →speed control levels
- Inertia control adjusts responsiveness with at least 8 levels → acceleration
- Emergency stop button.
- Error detection scheme on messages.

Requirements form

name	model train controller
purpose	control speed of <= 8 model trains
inputs	throttle, inertia, emergency stop, train #
outputs	train control signals
functions	set engine speed w. inertia; emergency stop
performance	can update train speed at least 10 times/sec
manufacturing cost	$50
power	wall powered
physical size/weight	console comfortable for 2 hands; < 2 lbs.

Taken/modified from "Computers as Components, Wolf"

Typical control sequence



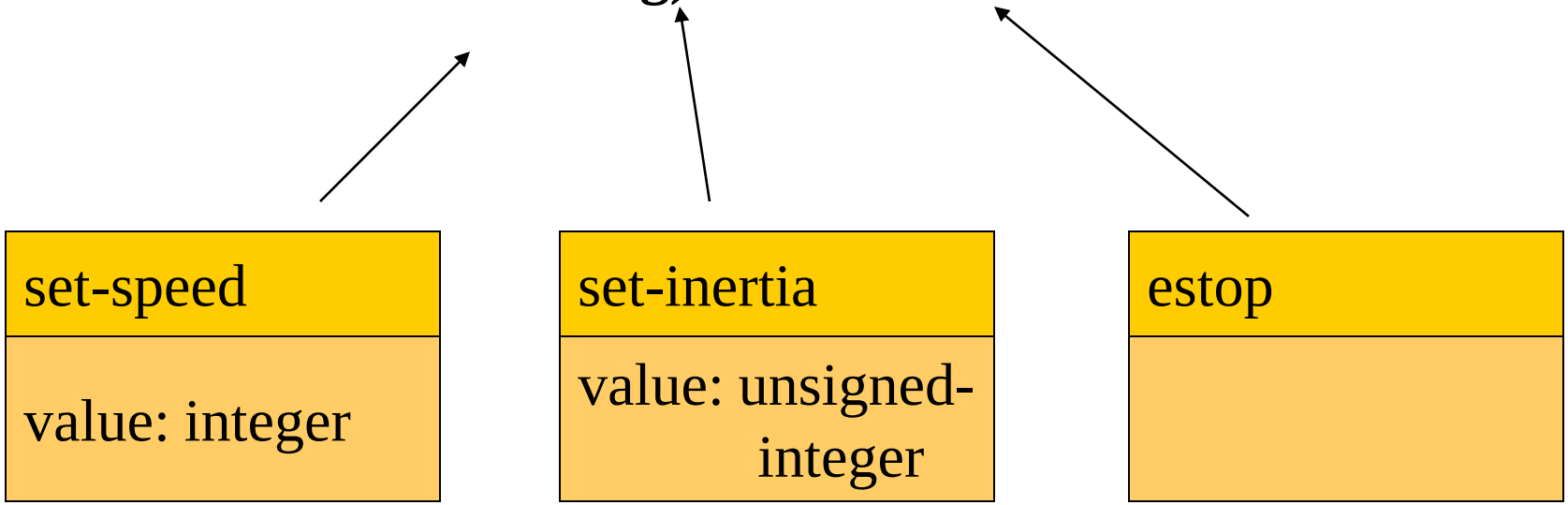
Taken/modified from "Computers as Components, Wolf"

Basic system commands

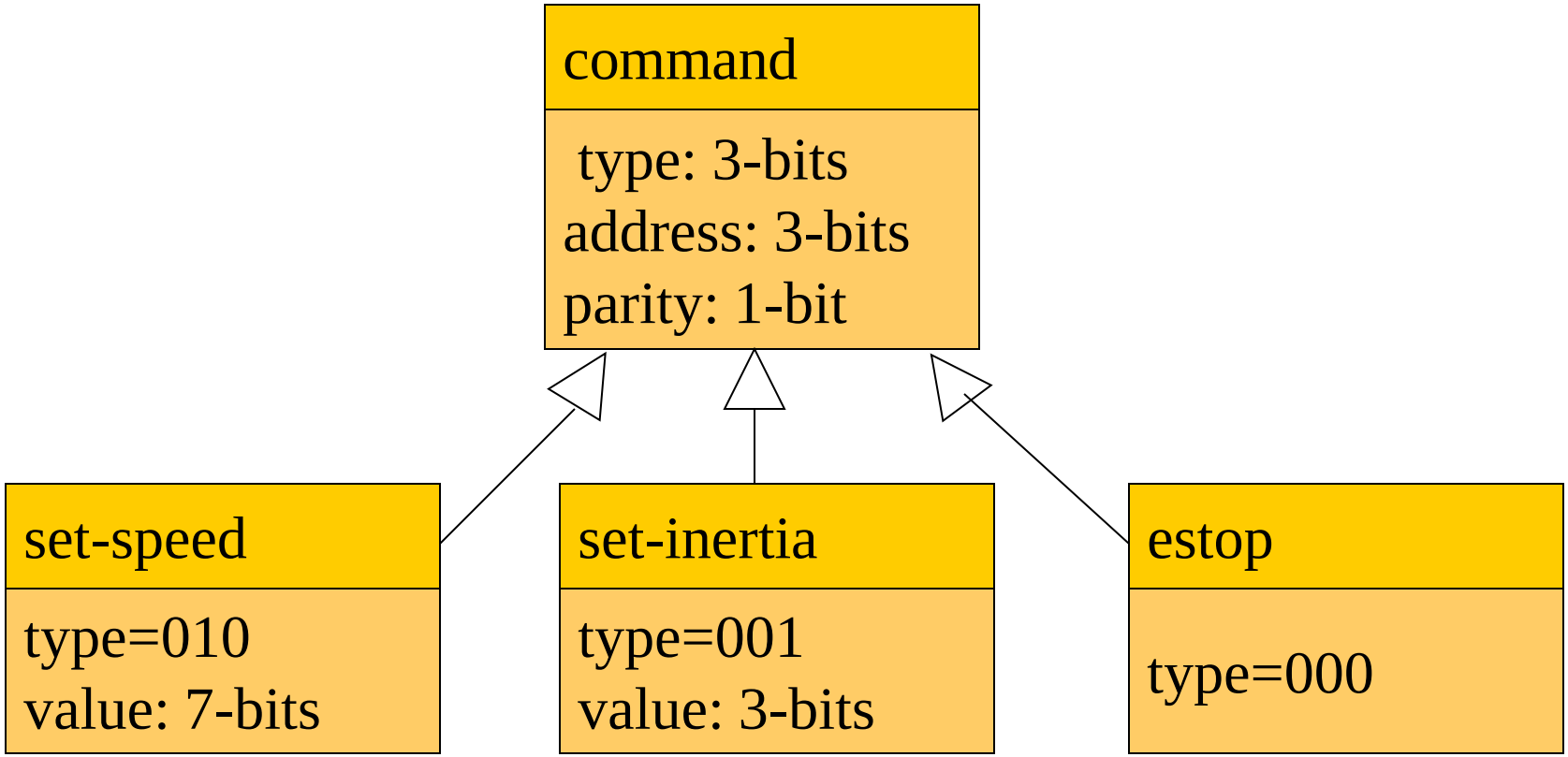
command name	parameters
set-speed	speed (positive/negative)
set-inertia	inertia-value (non-negative)
estop	none

Messages

Have common features: type, address to send data, parity (error checking)



Command class



Taken/modified from "Computers as Components, Wolf"

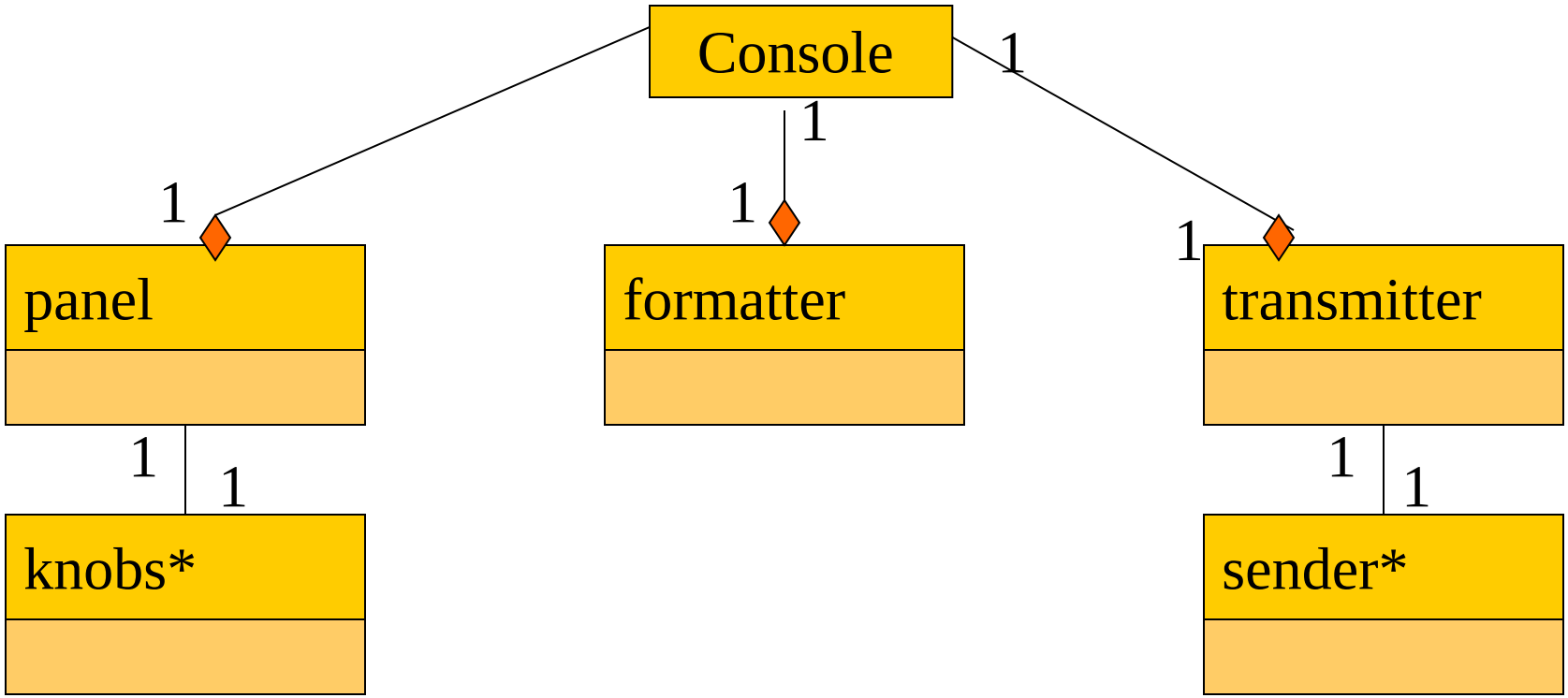
Major subsystem roles

- Console:
  - read state of front panel;
  - format messages;
  - transmit messages.
- Receiver:
  - receive message;
  - interpret message;
  - control the train.

System structure modeling

- Some classes define non-computer components.
  - Denoted by *name.
- Choose important systems at this point to show basic relationships.

Console system classes

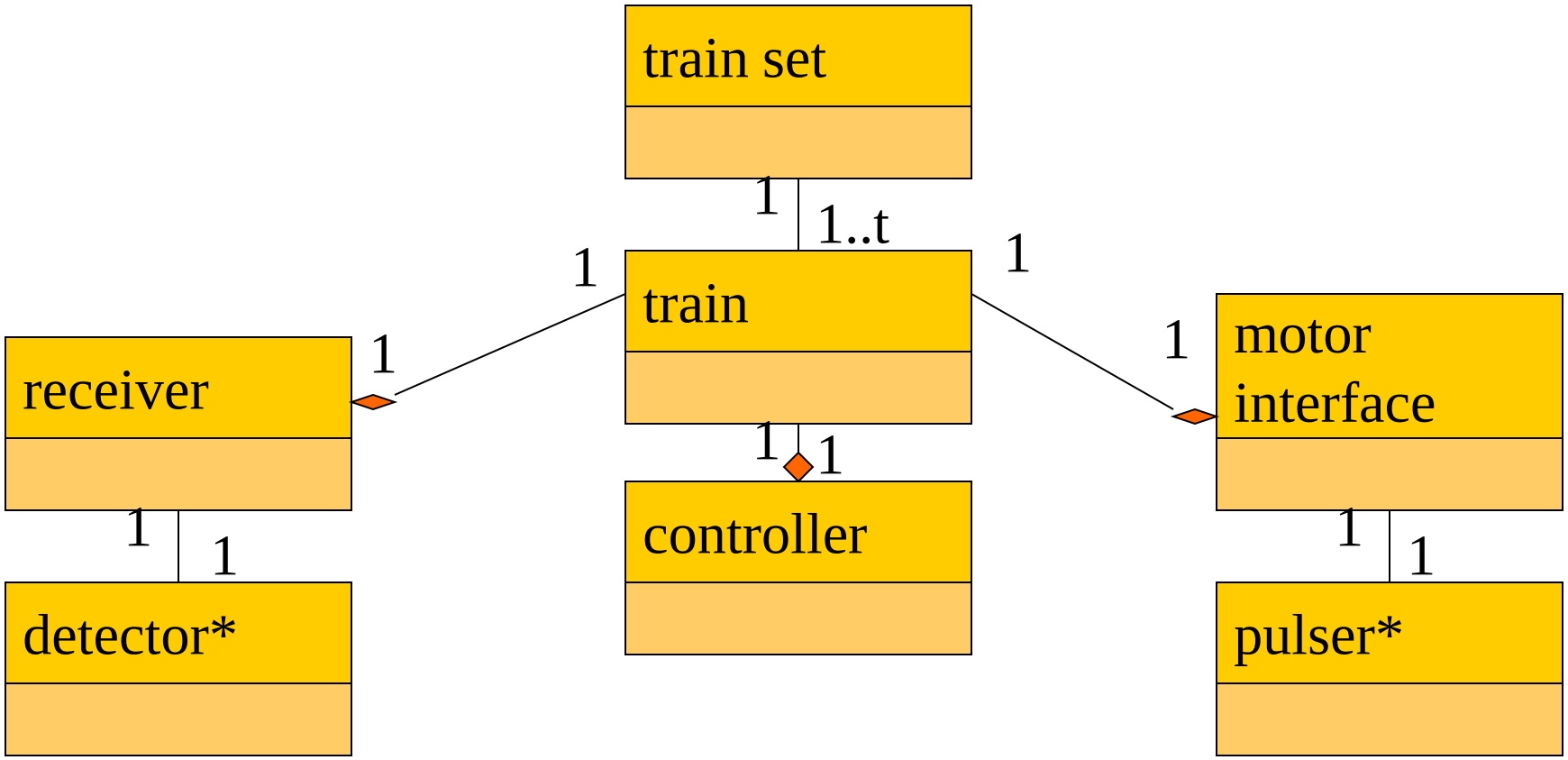


Taken/modified from "Computers as Components, Wolf"

Console class roles

- panel: describes analog knobs and interface hardware.
- formatter: turns knob settings into bit streams.
- transmitter: sends data on track.

Train system classes



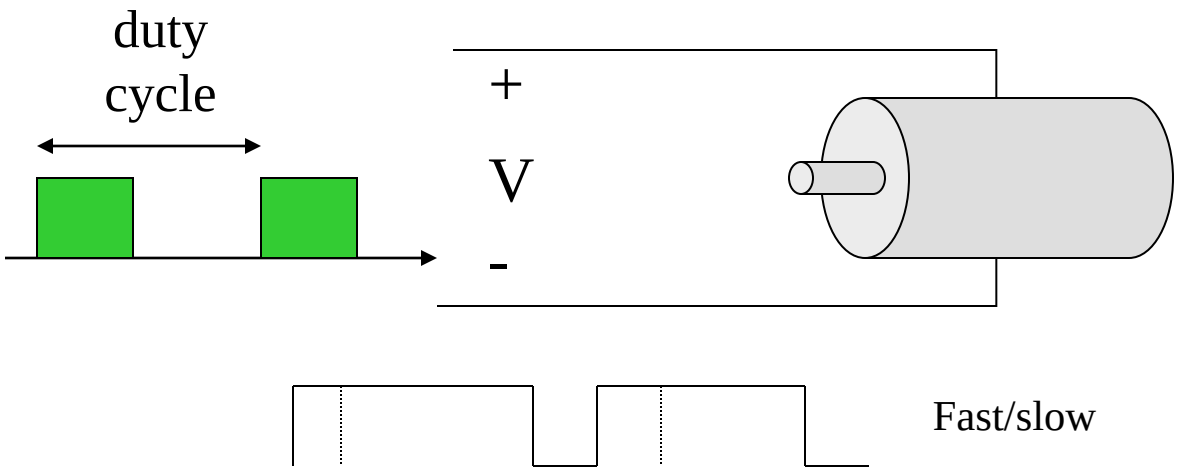
Taken/modified from "Computers as Components, Wolf"

Train class roles

- receiver: Digitizes signal from track.
- controller: Interprets received commands and makes control decisions.
- motor interface: Generates signals required by motor, e.g., to control SPEED, ACCELERATION

Train speed control

- Motor controlled by pulse width modulation:
- Sketching out the spec first helps us understand the basic relationships in the system.



Taken/modified from "Computers as Components, Wolf"

Detailed specification

- We can now fill in the details of the conceptual specification:
  - more classes;
  - behaviors.

Console/train: Physical objects *

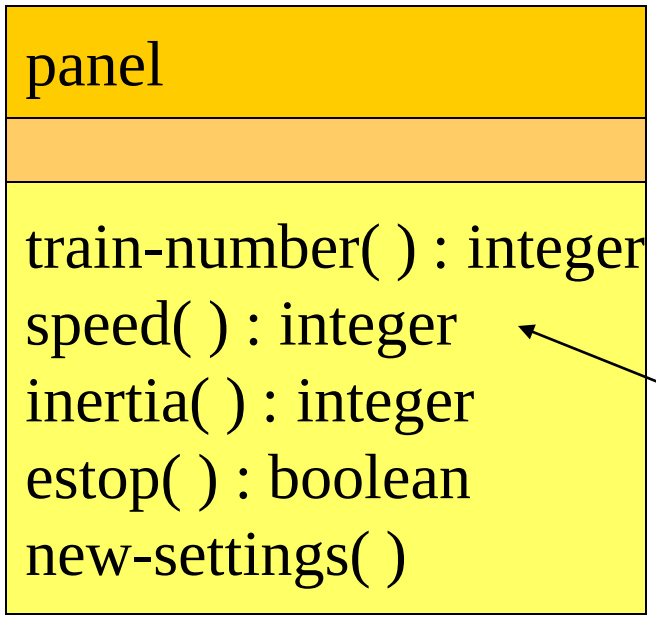
knobs*
train-knob: integer speed-knob inertia-knob emergency-stop: boolean

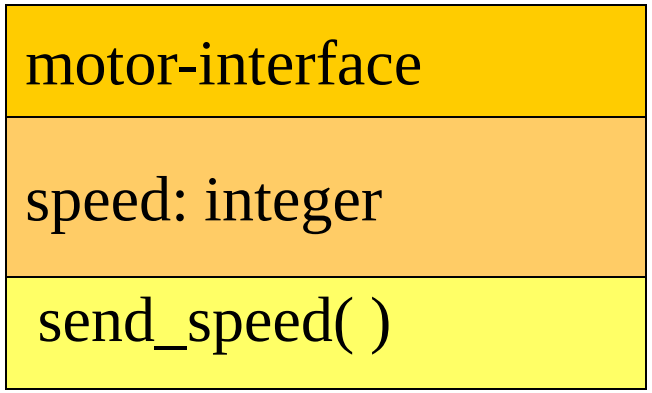
pulser*
pulse-width: unsigned-integer direction: boolean

sender*
send-bit( )

detector*
read-bit( )

Panel and motor interface classes



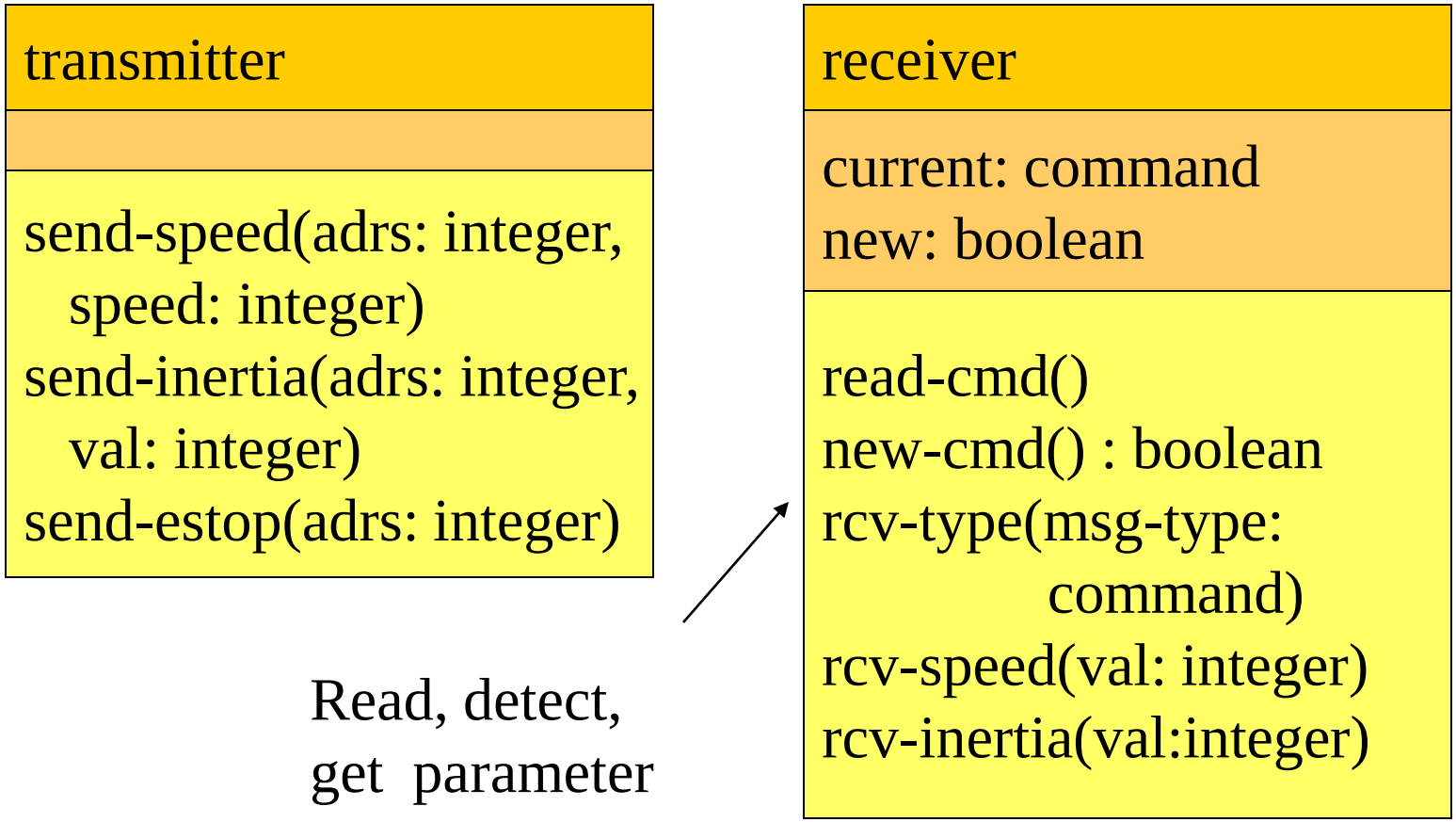


Reads from speed knob:

```
int speed( ){
  int spd;
  spd= inp(KNOBS_SPEED_PORT);
  return spd;
}
```

Taken/modified from "Computers as Components, Wolf"

Transmitter and receiver classes

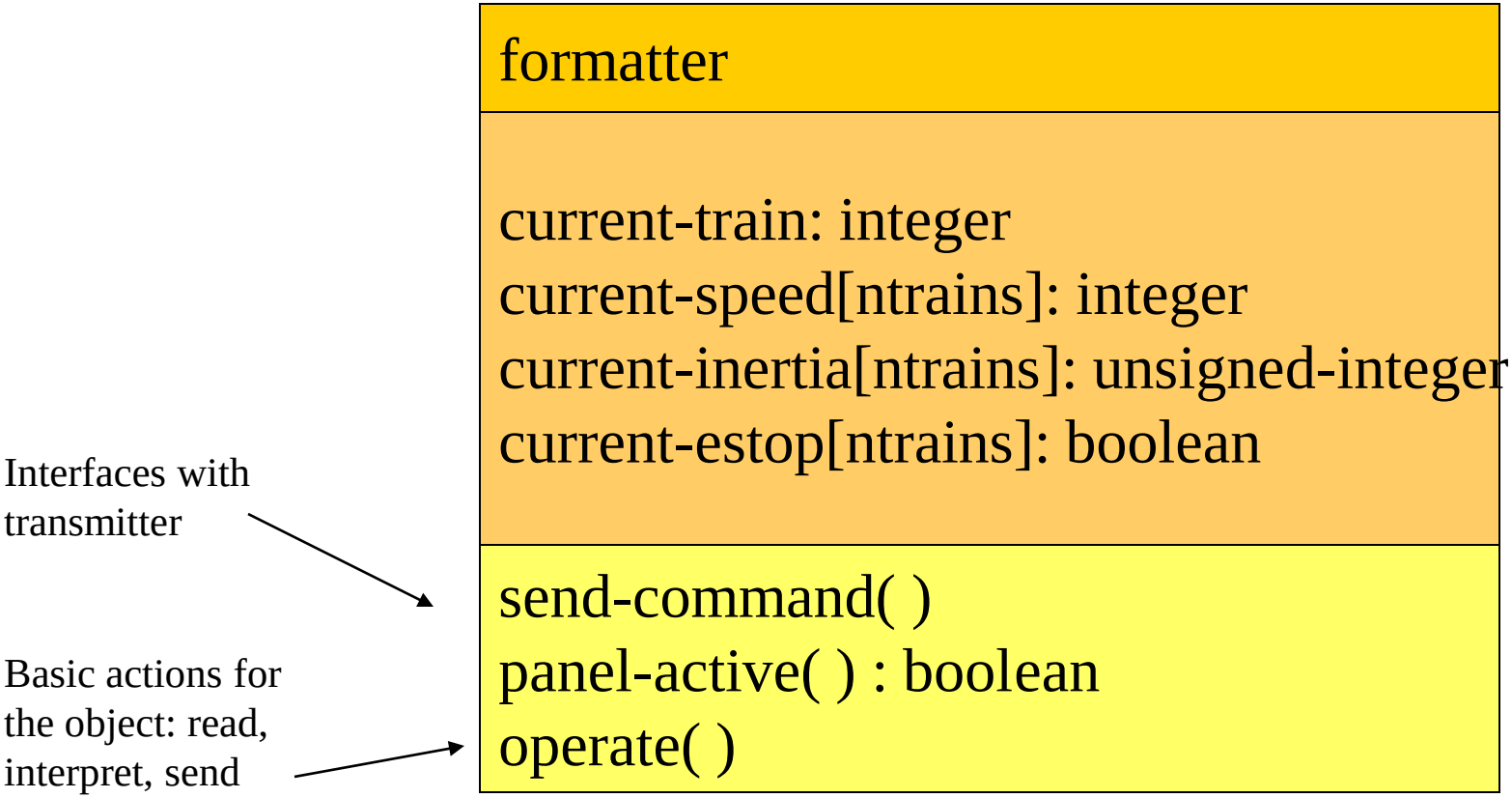


Taken/modified from "Computers as Components, Wolf"

Class descriptions

- transmitter class has one behavior (function) for each type of message sent.
- Receiver provides methods to:
  - detect a new message;
  - determine its type;
  - read its parameters (estop has no parameters).

Formatter class

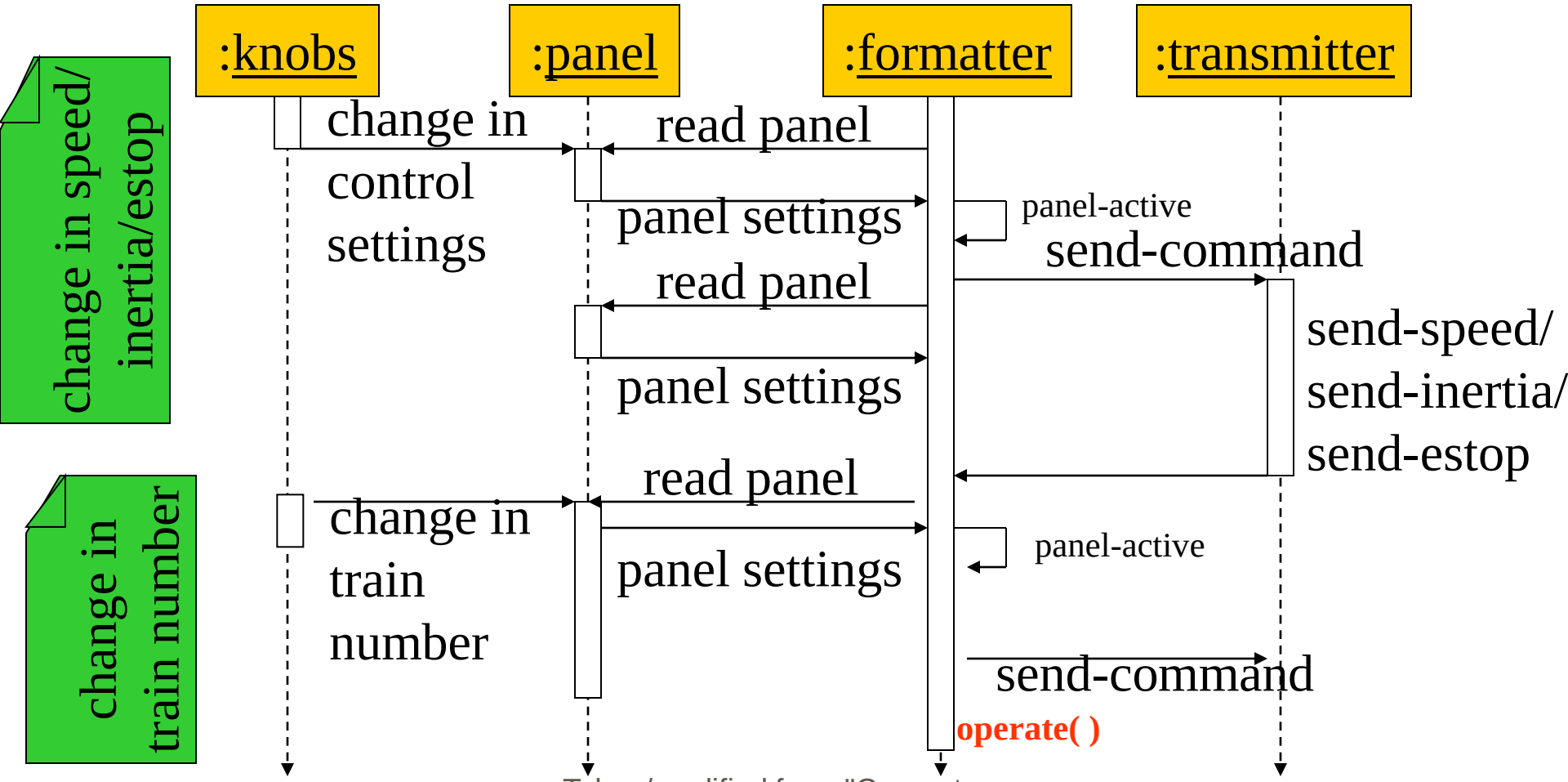


Taken/modified from "Computers as Components, Wolf"

Formatter class description

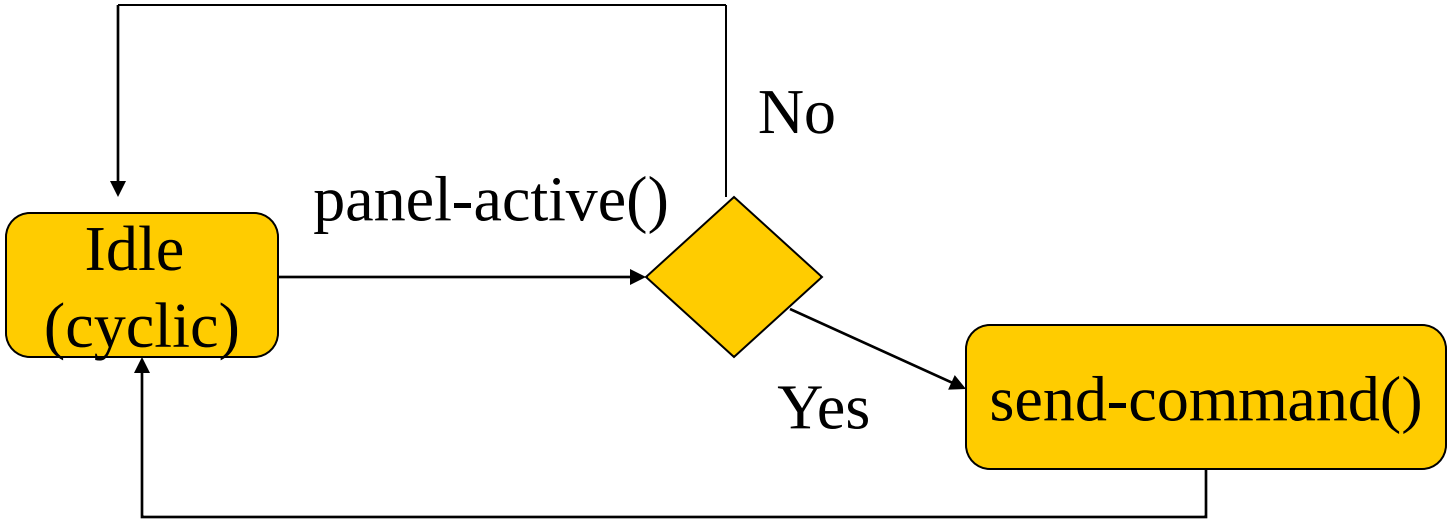
- Formatter class holds state for each train, setting for current train.
- The operate( ) function repeatedly performs the basic formatting task.

Control input sequence diagram



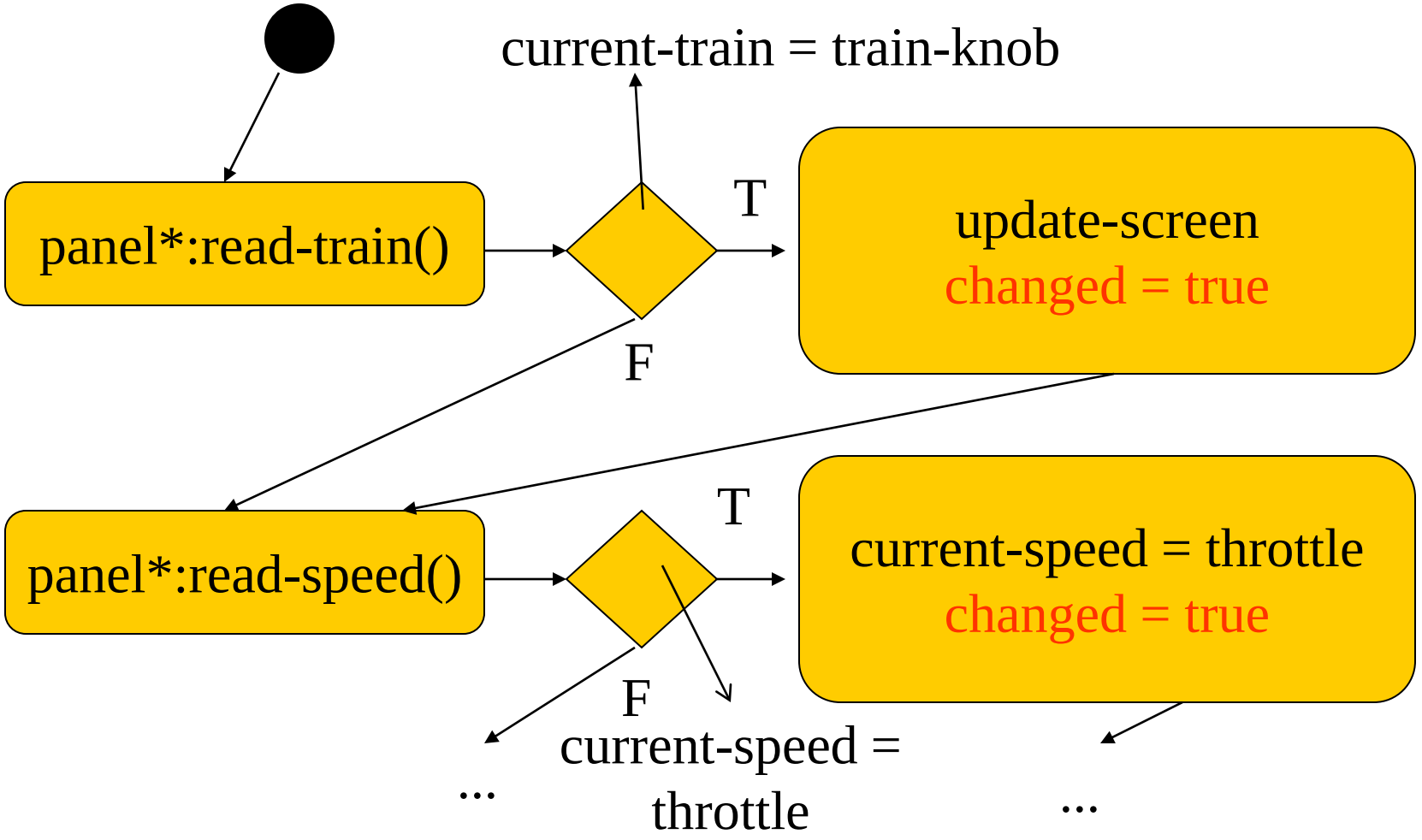
Taken/modified from "Computers as Components, Wolf"

Formatter operate behavior



Taken/modified from "Computers as Components, Wolf"

Panel-active behavior



Taken/modified from "Computers as Components, Wolf"

Controller class

Called by receiver when it gets a new command:

Interprets messages, issues a command

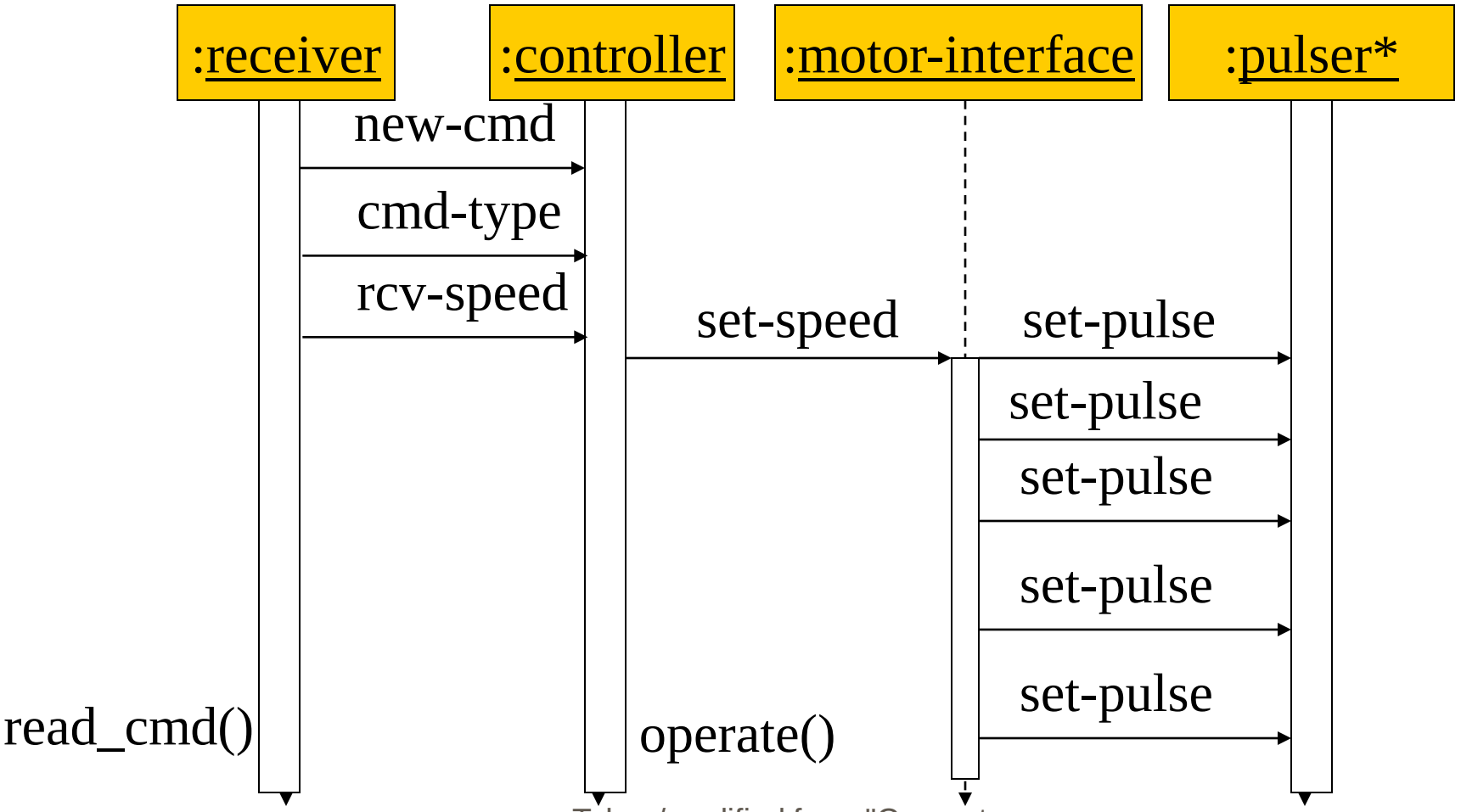
controller
current-train: integer current-speed[ntrains]: integer current-direction[ntrains]: boolean current-inertia[ntrains]: unsigned-integer
operate() issue-command()

Taken/modified from "Computers as Components, Wolf"

Setting the speed

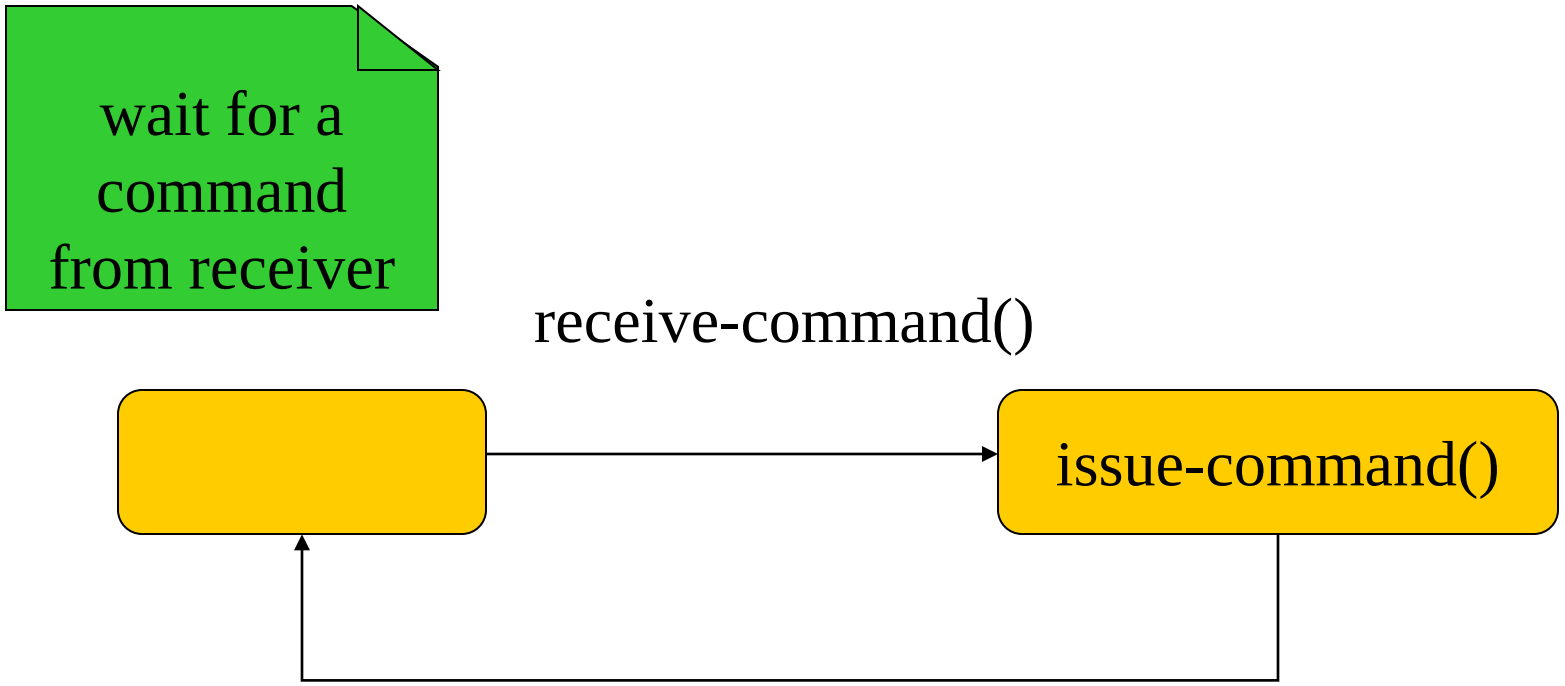
- Don’t want to change speed instantly.
- Controller should change speed gradually by sending several commands.

Sequence diagram for set-speed command



Taken/modified from "Computers as Components, Wolf"

Controller operate behavior



Summary

- Separate specification and programming.
  - Small mistakes are easier to fix in the spec.
  - Big mistakes in programming cost a lot of time.
- You can’t completely separate specification and architecture.
  - Make a few tasteful assumptions.

Assignment #1

- Chapter 1: Q1-3, Q1-4, Q1-5, Q1-9, Q1-15