Universal laws and architectures:

Theory and lessons from

nets, grids, brains, bugs,
planes, docs, fire, fashion,
art, turbulence, music, buildings,
cities, earthquakes, bodies, running, throwing,
Synesthesia, spacecraft, statistical mechanics
and zombies

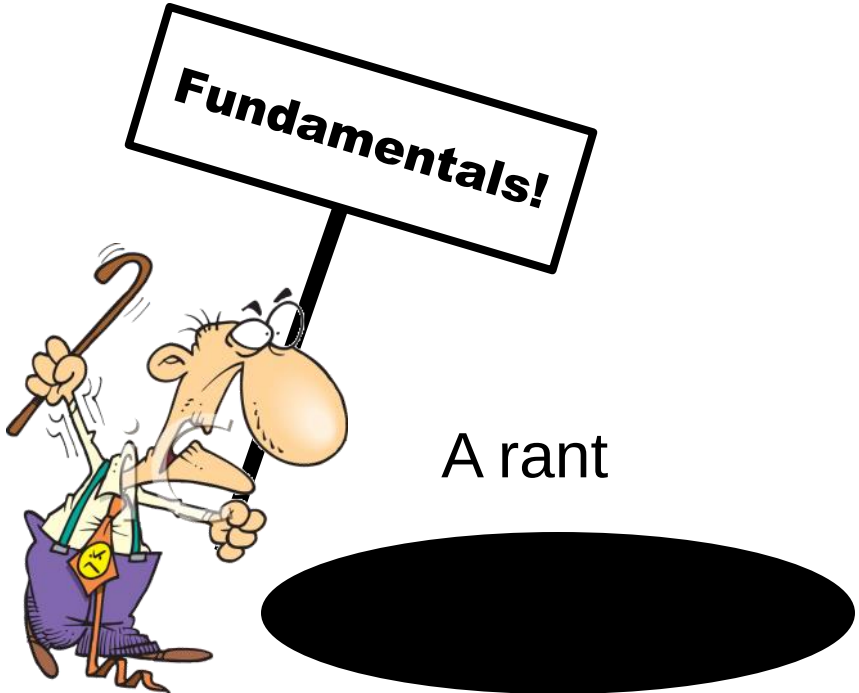
John Doyle 道陽

Jean-Lou Chameau Professor

Control and Dynamical Systems, EE, & BioE

Ca#1tech

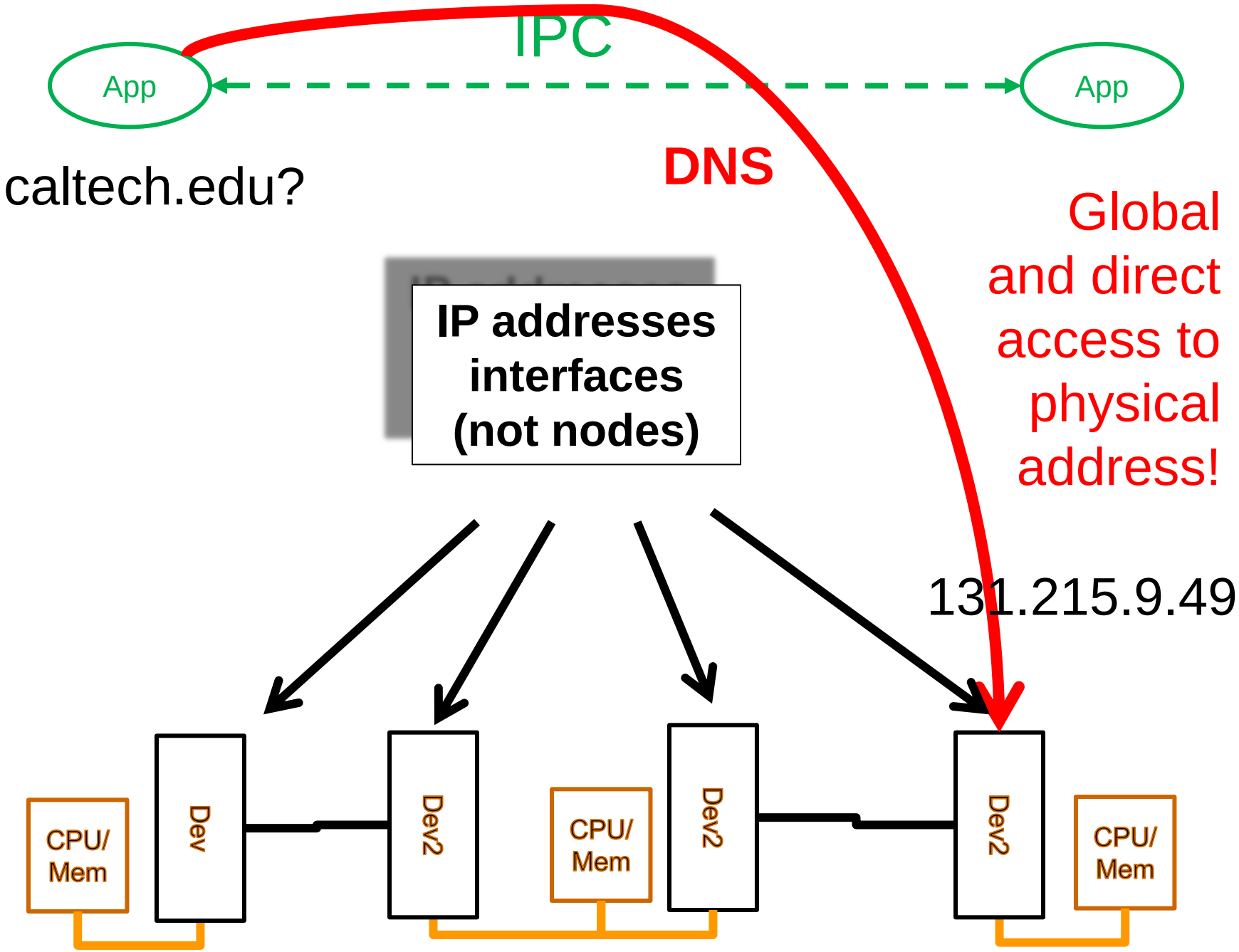
- Nets
- Grids (cyberphys)
- Brains
- Bugs (microbes)
- Medical physiology

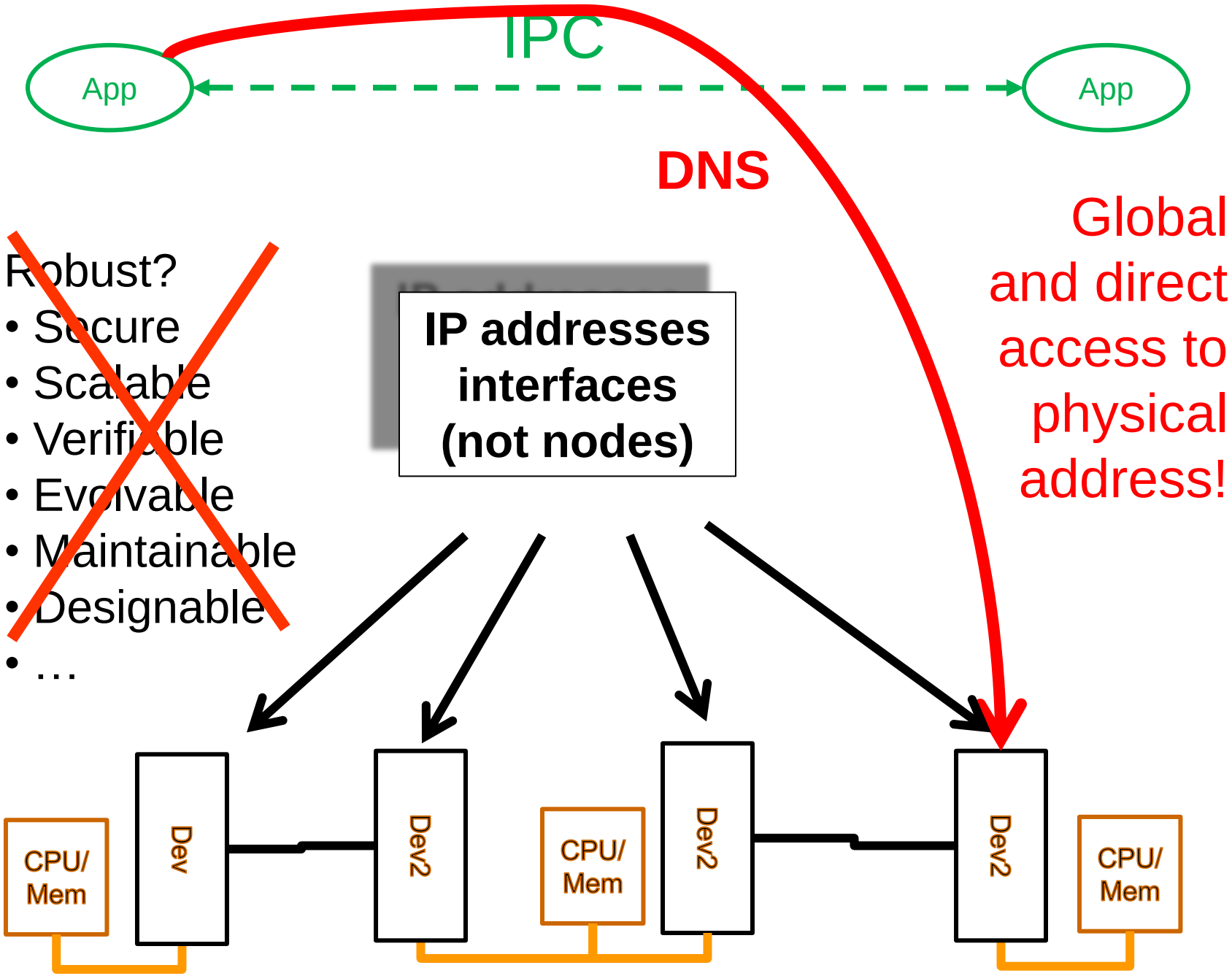


A rant

I {Case Study}

- Lots of aerospace
- Wildfire ecology
- Earthquakes
- Physics:
  - turbulence,
  - stat mech (QM?)
- “Toy”:
  - Lego
  - clothing, fashion
- Buildings, cities
- Ants
- **Synesthesia**


IPC
App
App
DNS
caltech.edu?
Global and direct access to physical address!
IP addresses interfaces (not nodes)
131.215.9.49
CPU/ Mem
Dev
Dev2
CPU/ Mem
Dev2
Dev2
CPU/ Mem


IPC
App
App
DNS
Global and direct access to physical address!
Robust?
• Secure
• Scalable
• Verifiable
• Evolvable
• Maintainable
• Designable
• …
IP addresses interfaces (not nodes)
CPU/ Mem
Dev
Dev2
CPU/ Mem
Dev2
Dev2
CPU/ Mem

Requirements on systems and architectures

accessible
accountable
accurate
adaptable
administrable
affordable
auditable
autonomy
available
credible
process
capable
compatible
composable
configurable
correctness
customizable
debugable
degradable
determinable
demonstrable

dependable
deployable
discoverable
distributable
durable
effective
efficient
evolvable
extensible
fail transparent
fast
fault-tolerant
fidelity
flexible
inspectable
installable
Integrity
interchangeable
interoperable
learnable
maintainable

manageable
mobile
modifiable
modular
nomadic
operable
orthogonality
portable
precision
predictable
producible
provable
recoverable
relevant
reliable
repeatable
reproducible
resilient
responsive
reusable
robust

safety
scalable
seamless
self-sustainable
serviceable
supportable
securable
simplicity
stable
standards
compliant
survivable
sustainable
tailorable
testable
timely
traceable
ubiquitous
understandable
upgradable
usable

Sustainable ≈ robust + efficient

accessible
accountable
accurate
adaptable
administrable
affordable
auditable
autonomy
available
compatible
composable
configurable
correctness
customizable
debugable
degradable
determinable
demonstrable

dependable
deployable
discoverable
distributable
durable
effective
efficient
evolvable
extensible
fail transparent
fast
fault-tolerant
fidelity
flexible
inspectable
installable
Integrity
interchangeable
interoperable
learnable
maintainable

manageable
mobile
modifiable
modular
nomadic
operable
orthogonality
portable
precision
predictable
producible
provable
recoverable
relevant
reliable
repeatable
reproducible
resilient
responsive
reusable
robust

safety
scalable
seamless
self-sustainable
serviceable
supportable
securable
simple
stable
standards
survivable
sustainable
tailorable
testable
timely
traceable
ubiquitous
understandable
upgradable
usable

PCA ≈ Principal ***Concept*** Analysis ☺

accessible
accountable
accurate
adaptable
adminis
afforda
auditab
autono
availab
compat
compos
configu
correct
custom
debuga
degrad
determ
demons

dependable
deployable
discoverable
distributable

manageable
mobile
modifiable
modular

safety
scalable
seamless
self-sustainable
eable
rtable
able
e
e
ards
able
inable
able
ble
y
able
itous
standable
dable
e

fragile

Simple
dichotomous
tradeoff pairs

robust

efficient

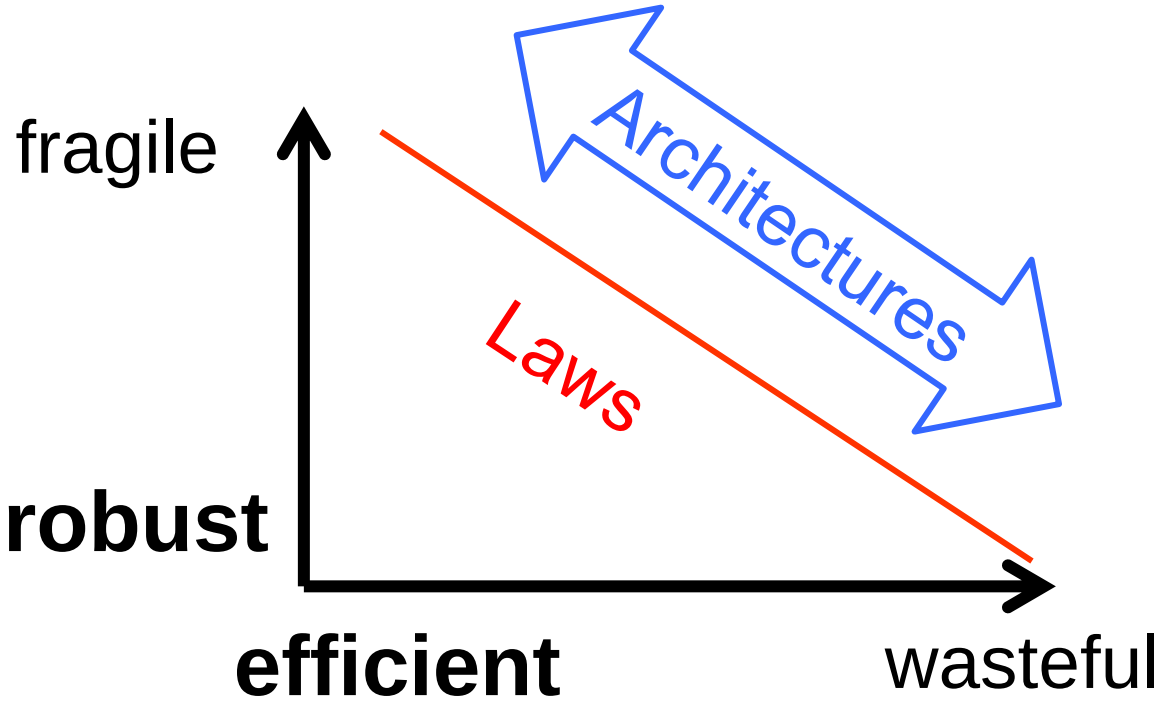
wasteful

interoperable
learnable
maintainable

reusable

robust

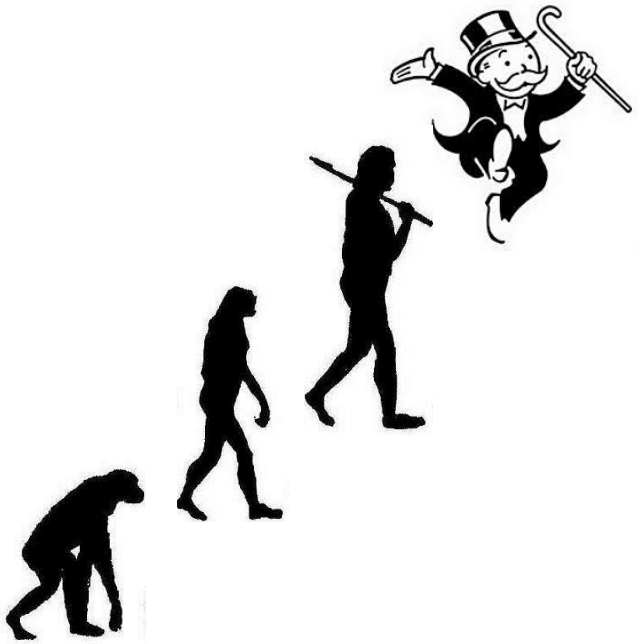
Layering Feedback



Layers/feedback in biology

Working backwards

- Money/finance/lobbyists/etc
- Society/agriculture/weapons/etc
- Bipedalism
- Maternal care
- Warm blood
- Flight
- Mitochondria
- Translation (ribosomes)
- Glycolysis (see 2011 Science paper)



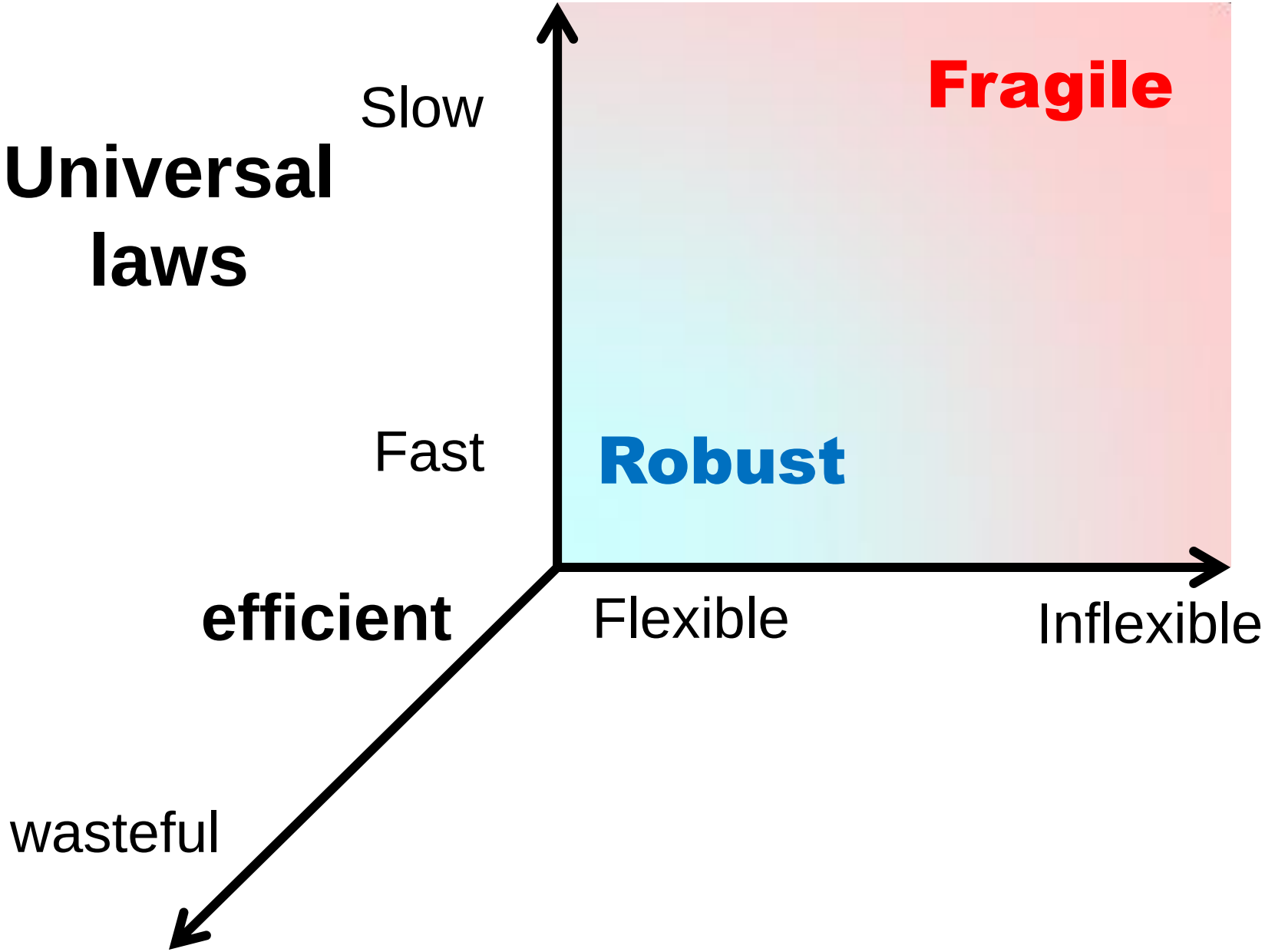
Universal laws

Fragile

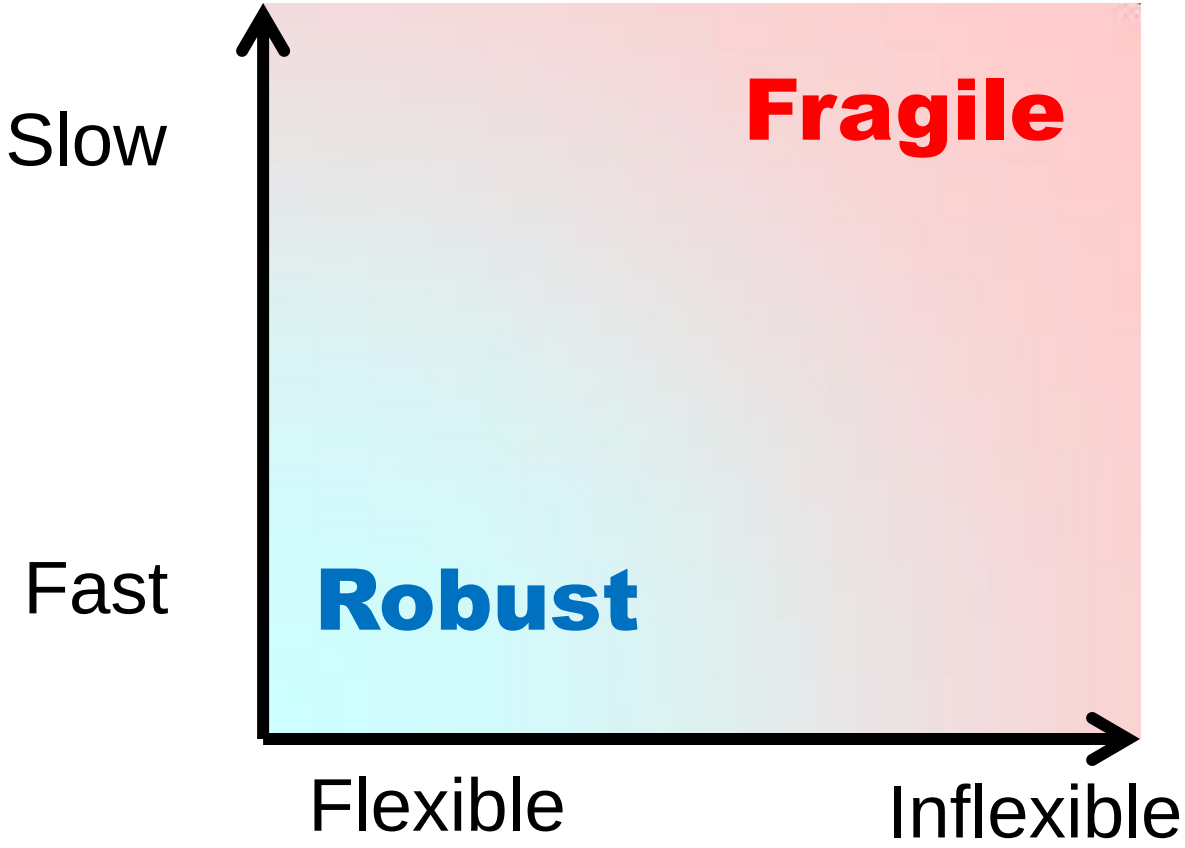
Robust

efficient

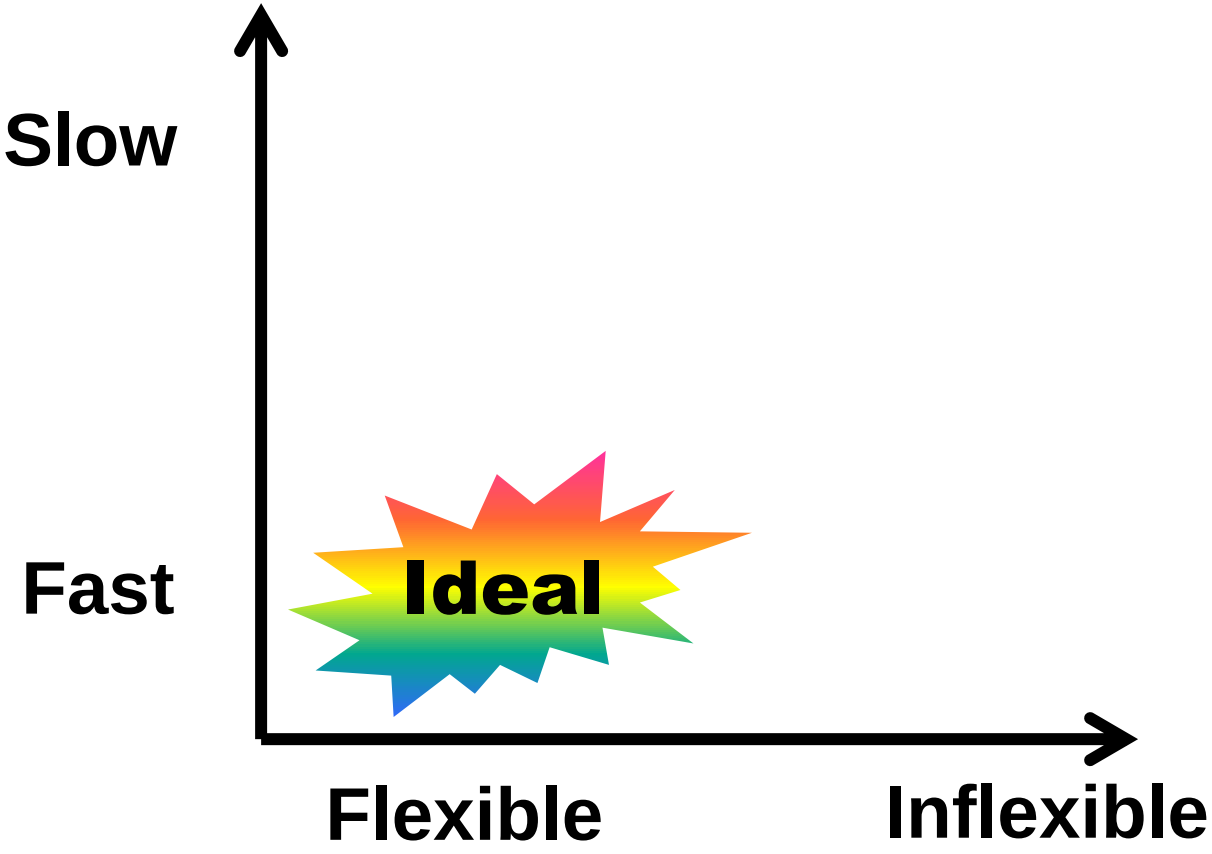
wasteful

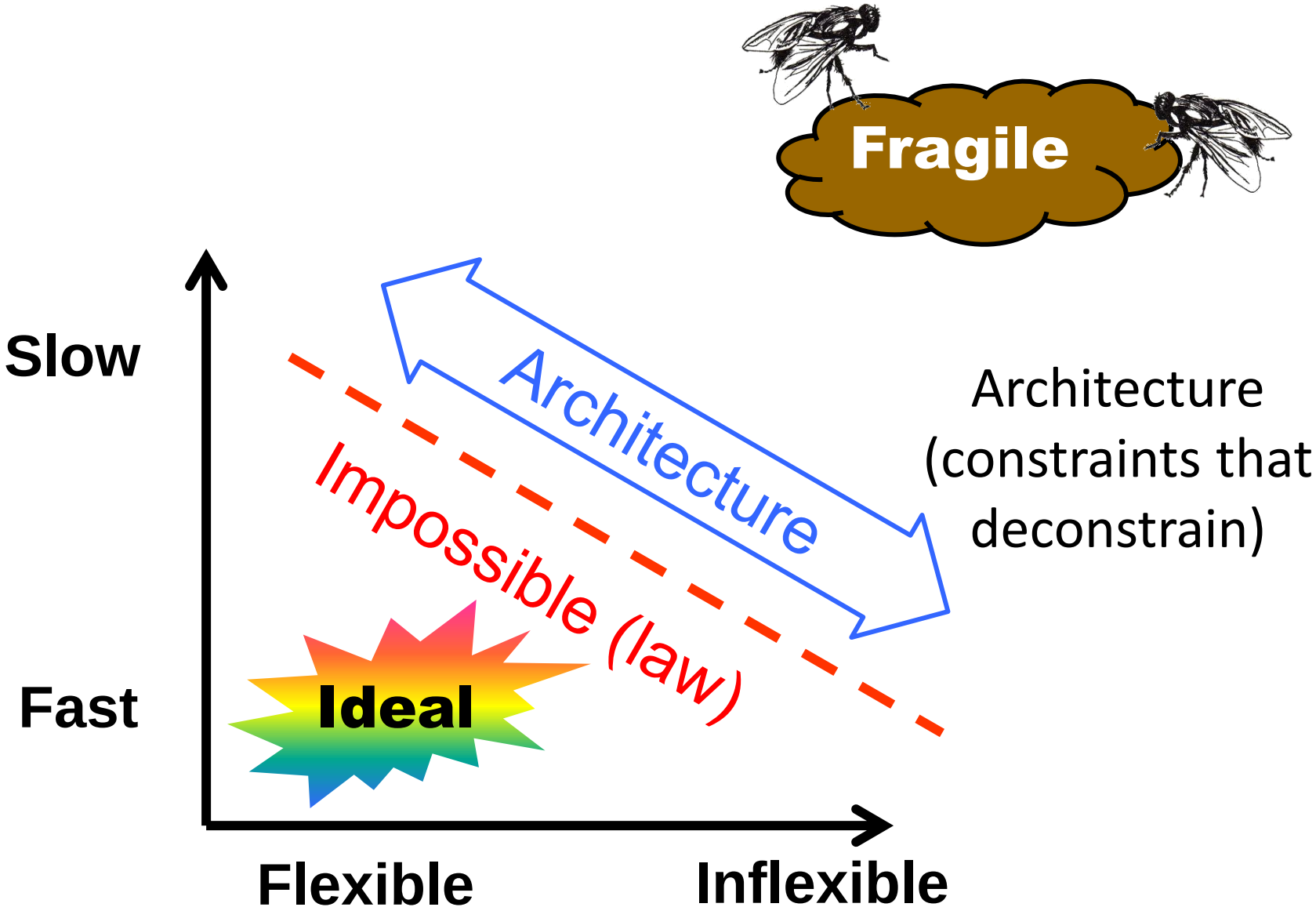

Universal laws
Slow
Fast
Fragile
Robust
efficient
Flexible
Inflexible
wasteful

Universal laws

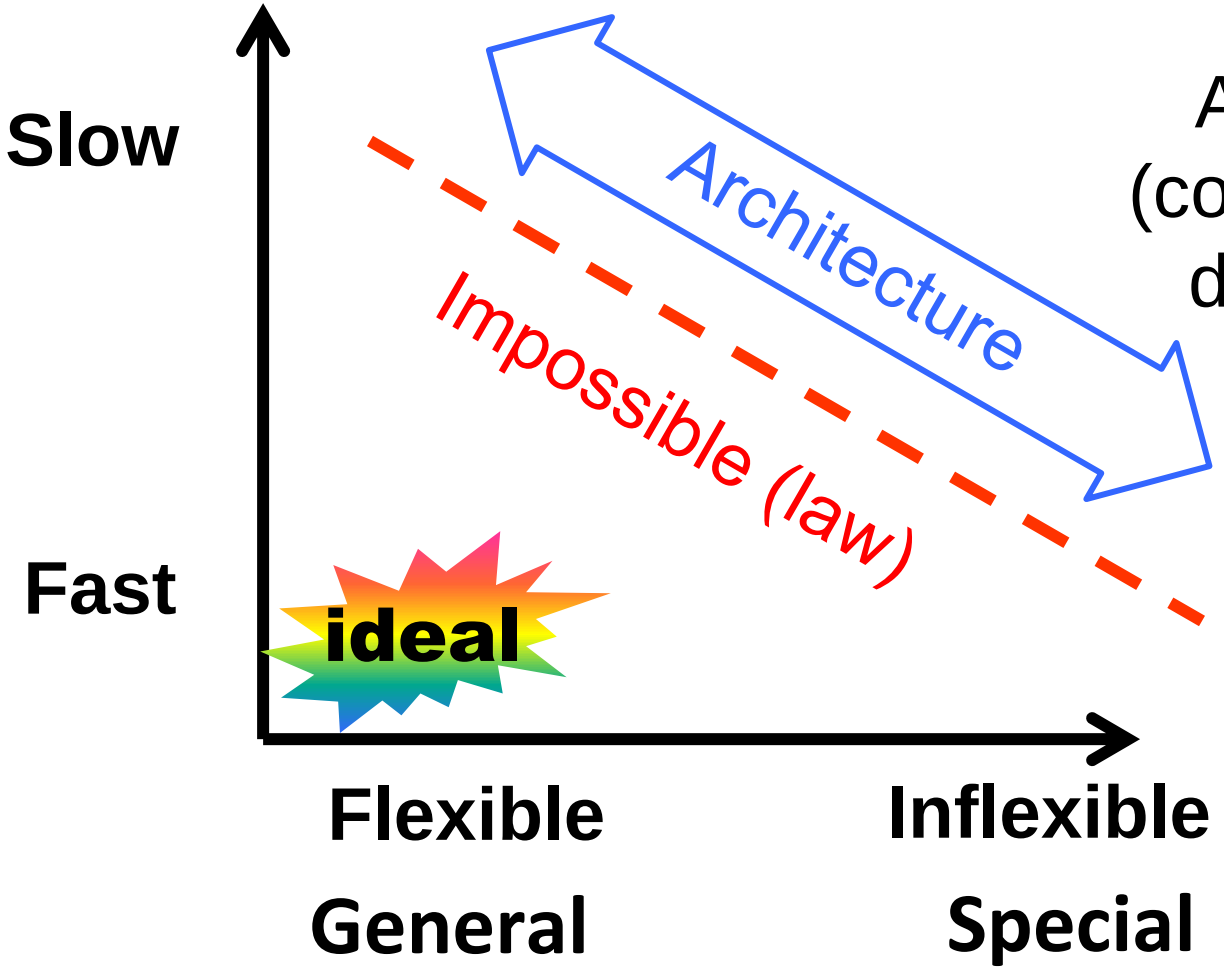


Architecture?




Fragile
Slow
Fast
Architecture
Impossible (law)
Ideal
Flexible
Inflexible
Architecture
(constraints that
deconstrain)

Universal laws and architectures (Turing)



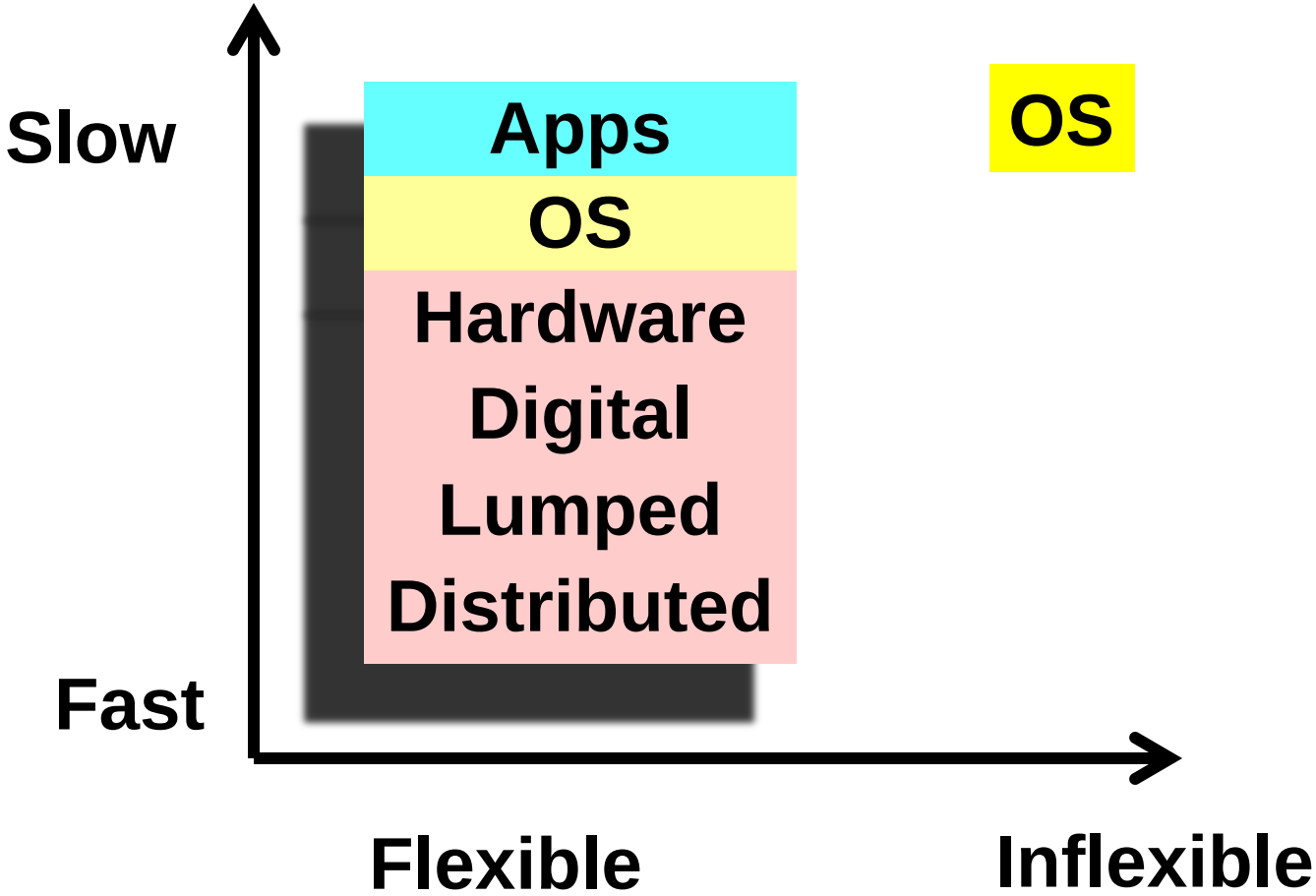
Architecture (constraints that deconstrain)

Tech implications/extensions

Apps

OS

HW



Layered Architecture

Apps

OS

Hardware

Digital

Lumped

Distributed

Any layer needs all lower layers.

Layered architectures 101

Apps

OS

Hardware

Digital

Lumped

Distributed

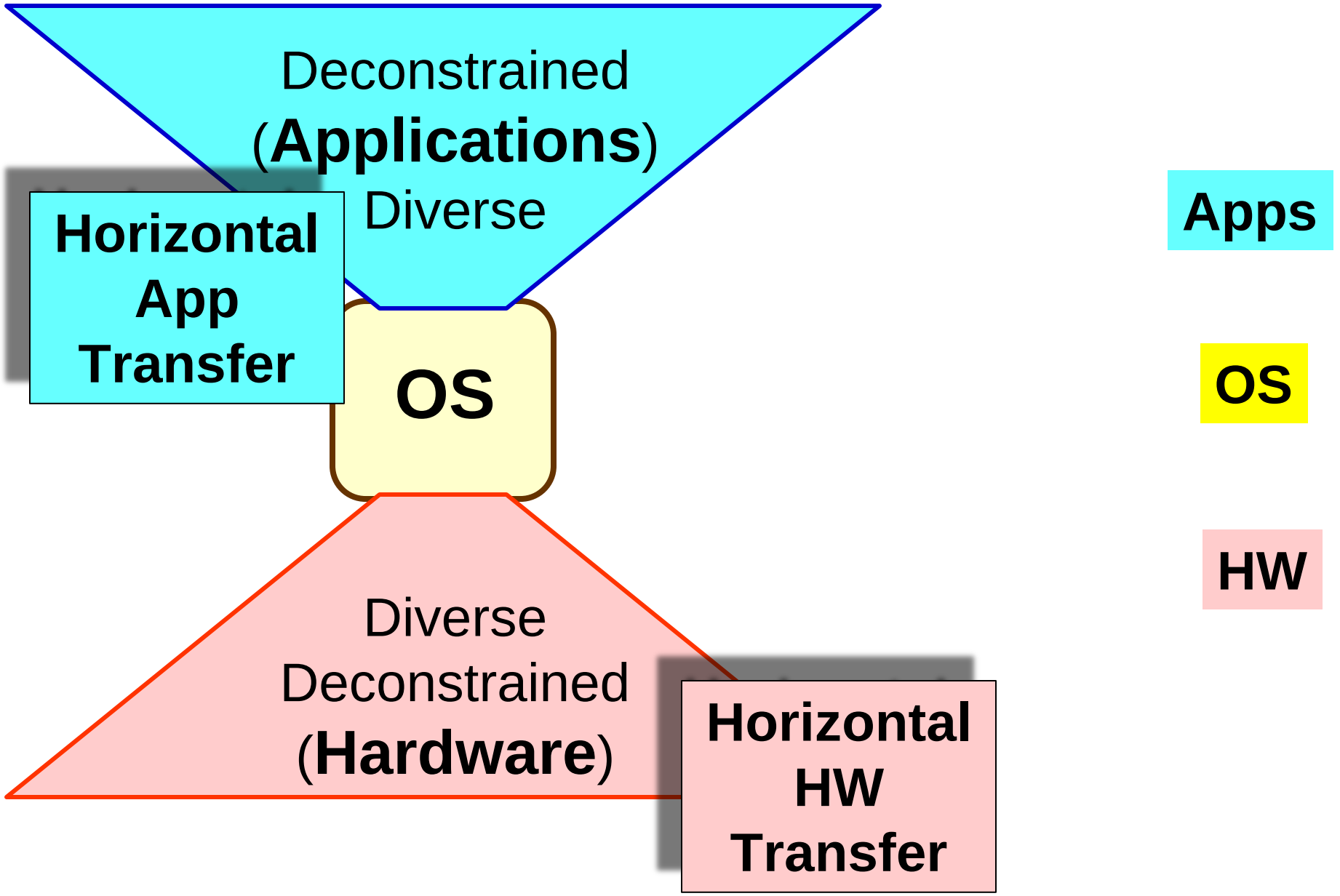
Operating System

Apps

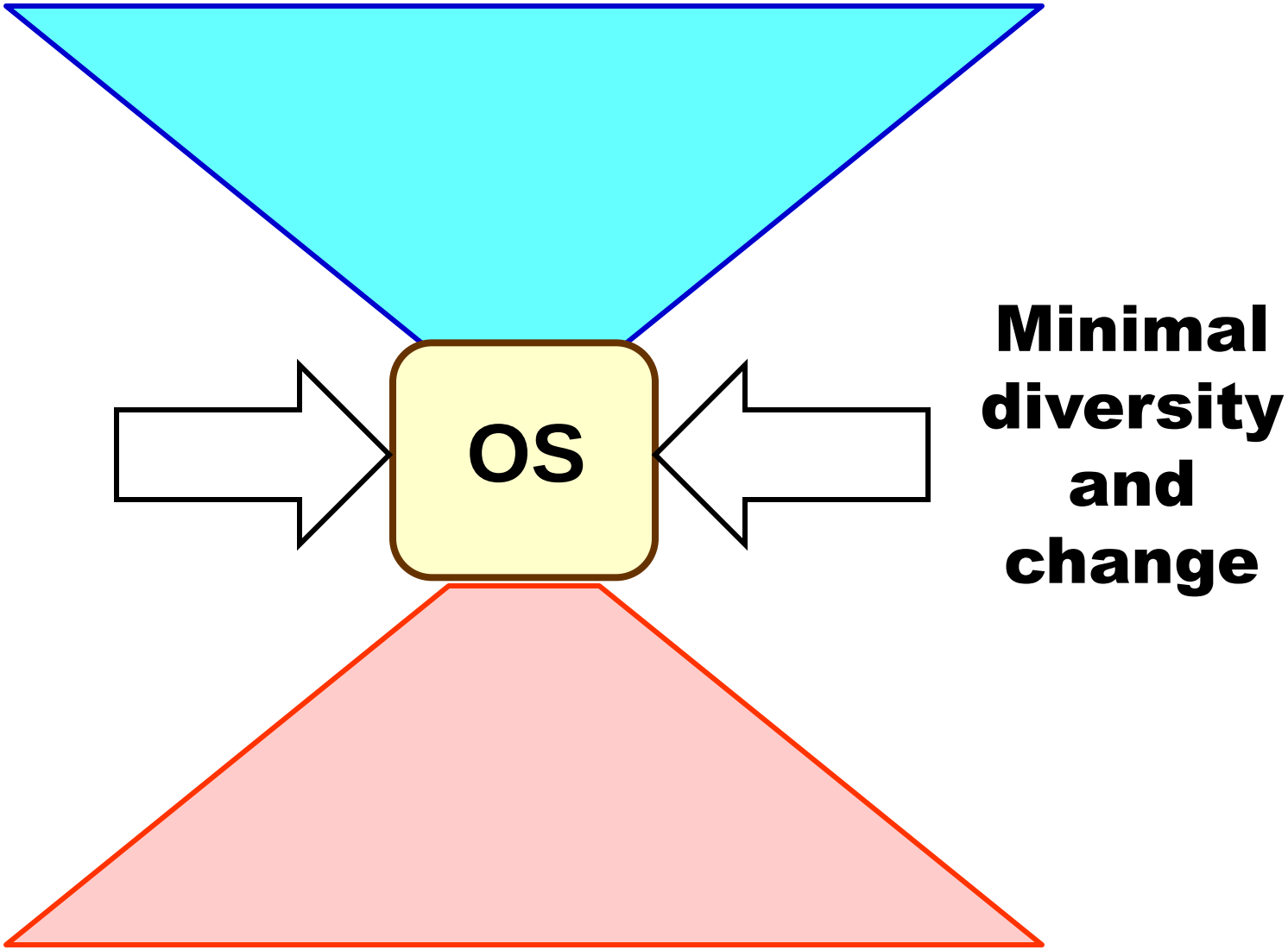
OS

HW

Layered architectures



Layered architectures

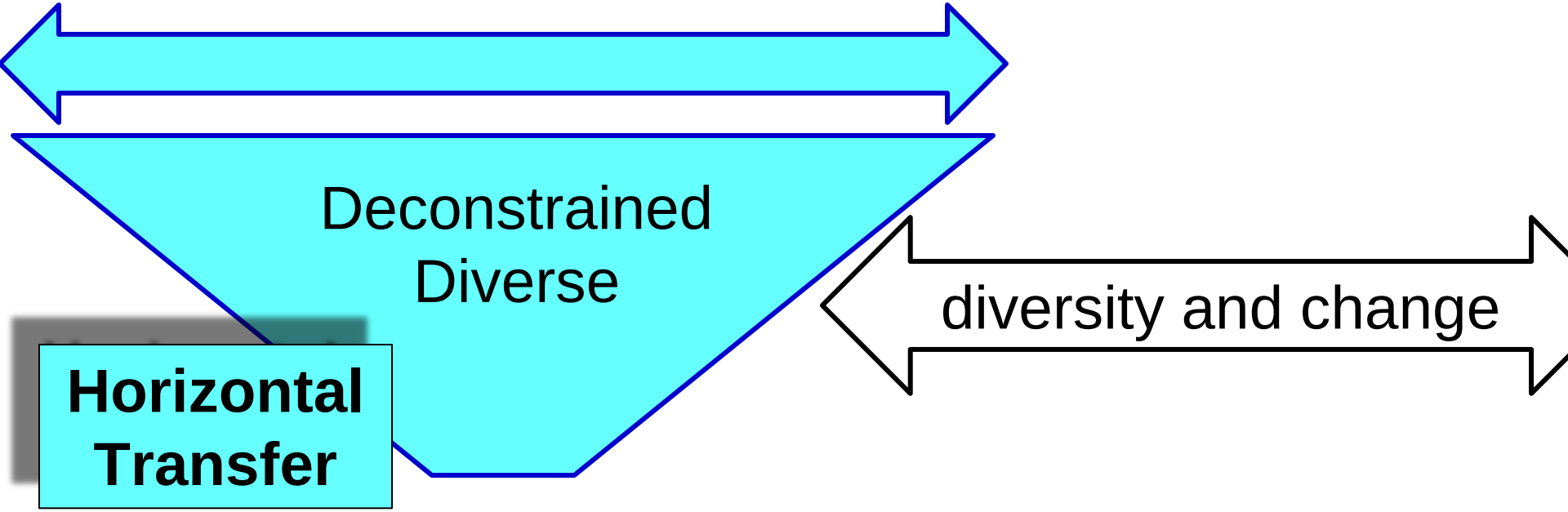


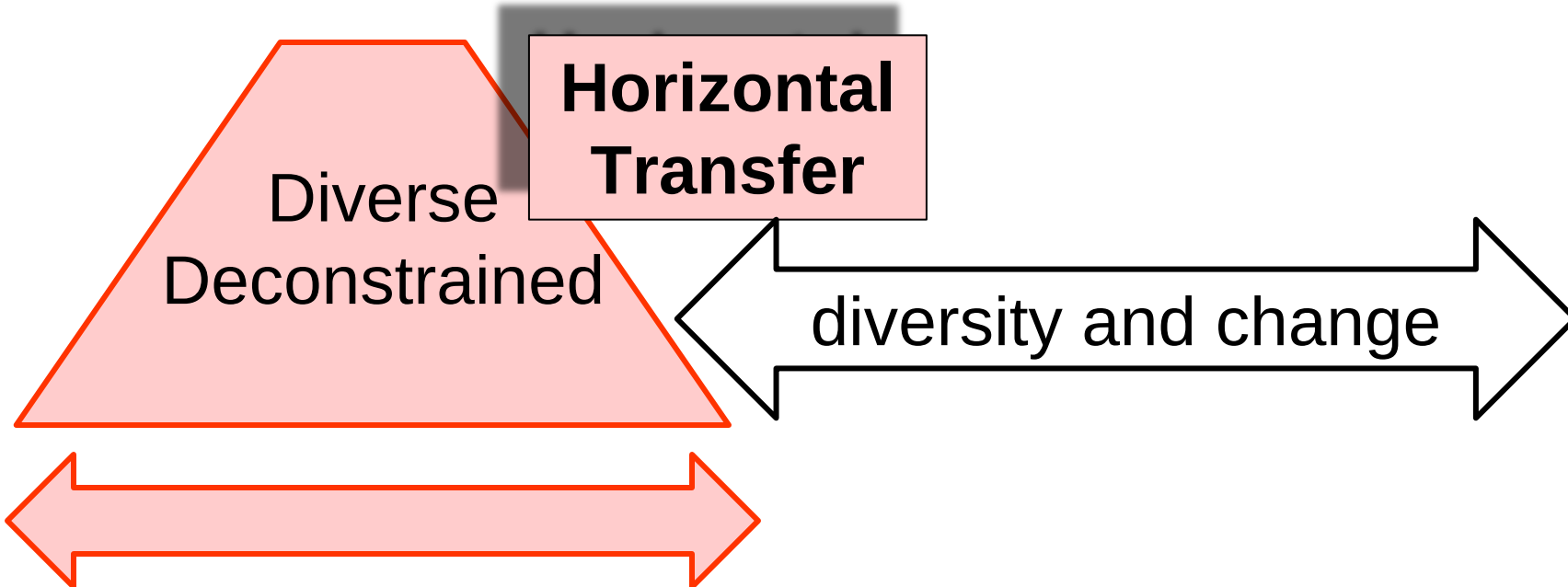
Minimal diversity and change

Apps

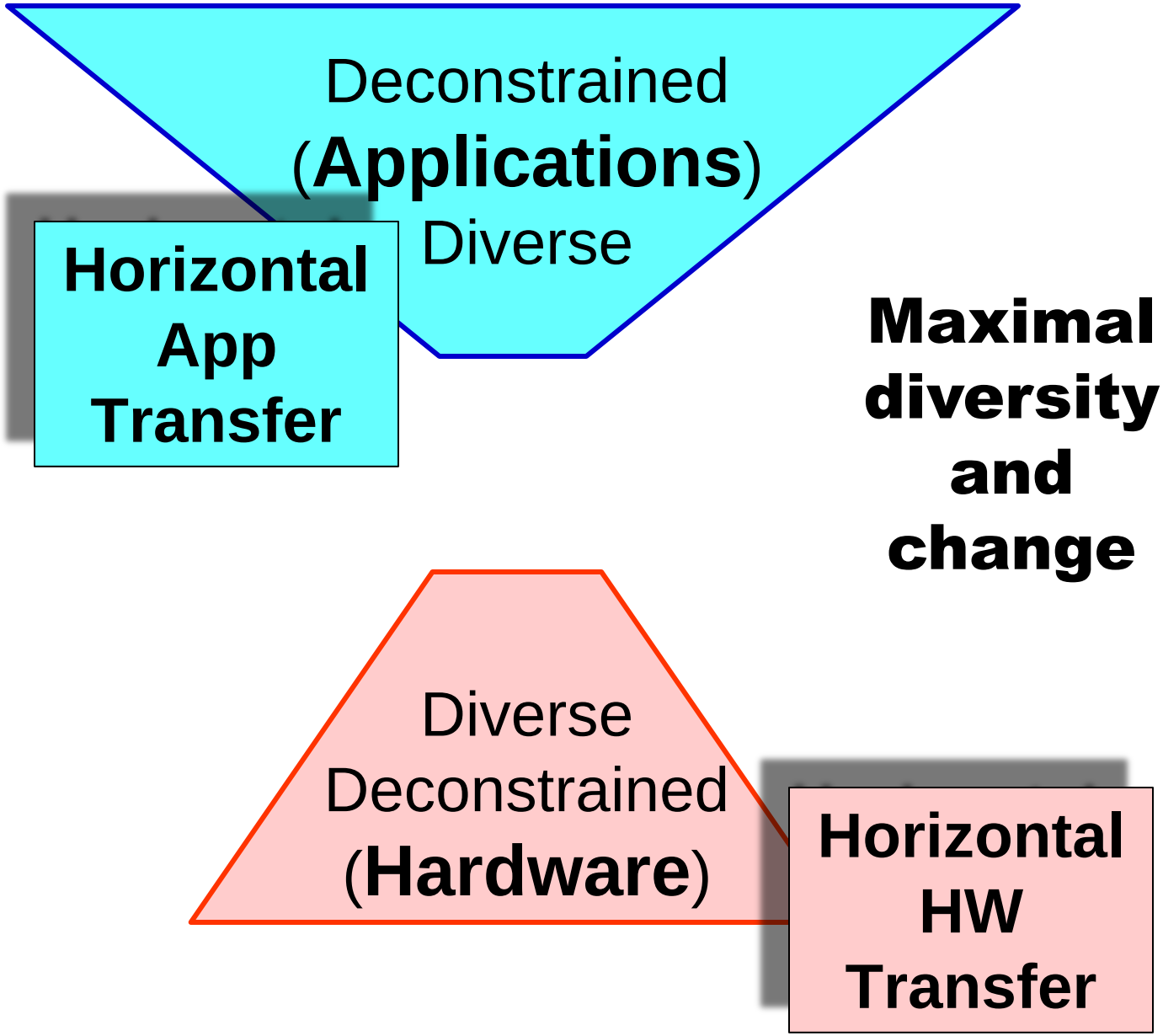
OS

HW


Deconstrained
Diverse
diversity and change
Horizontal
Transfer


Horizontal
Transfer
Diverse
Deconstrained
diversity and change

Layered architectures

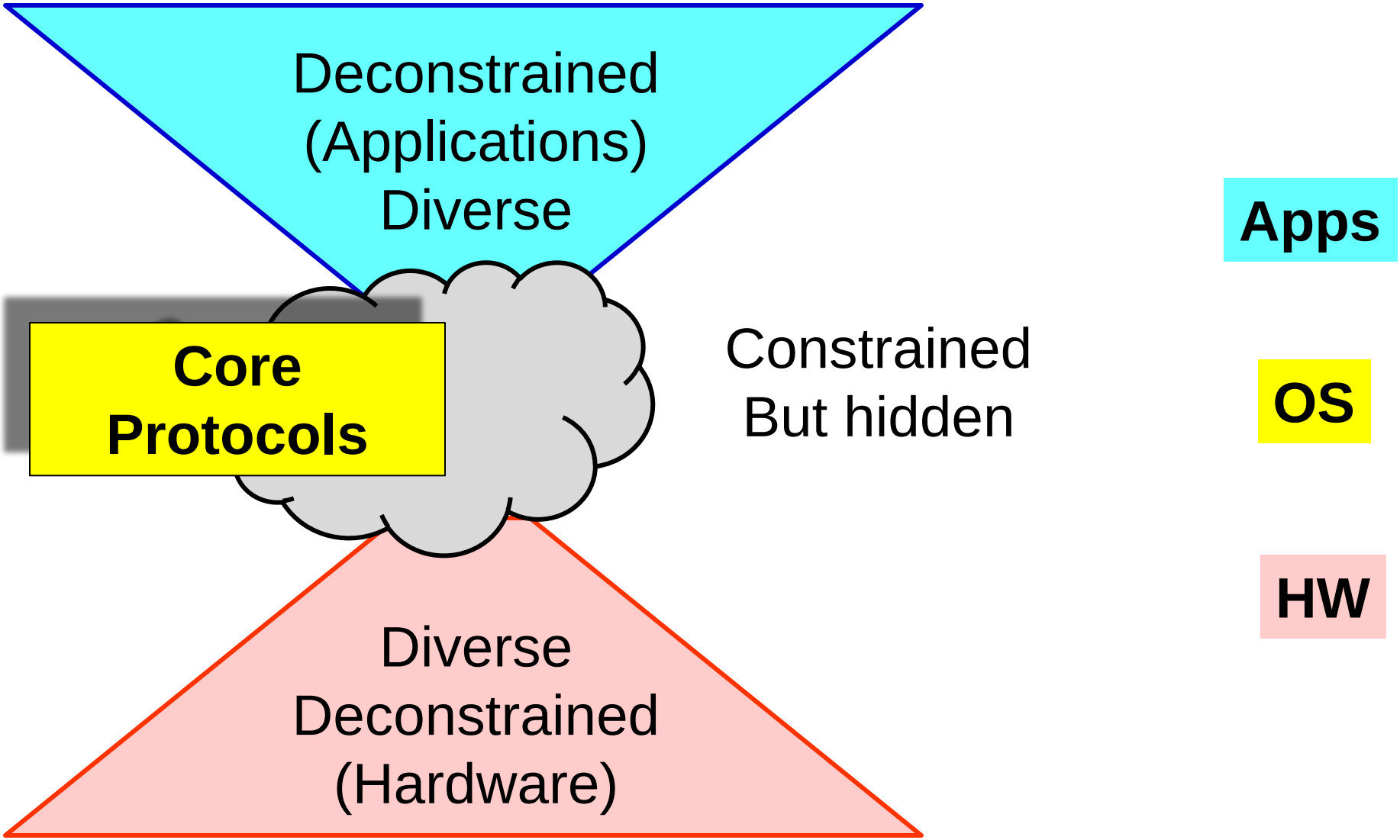


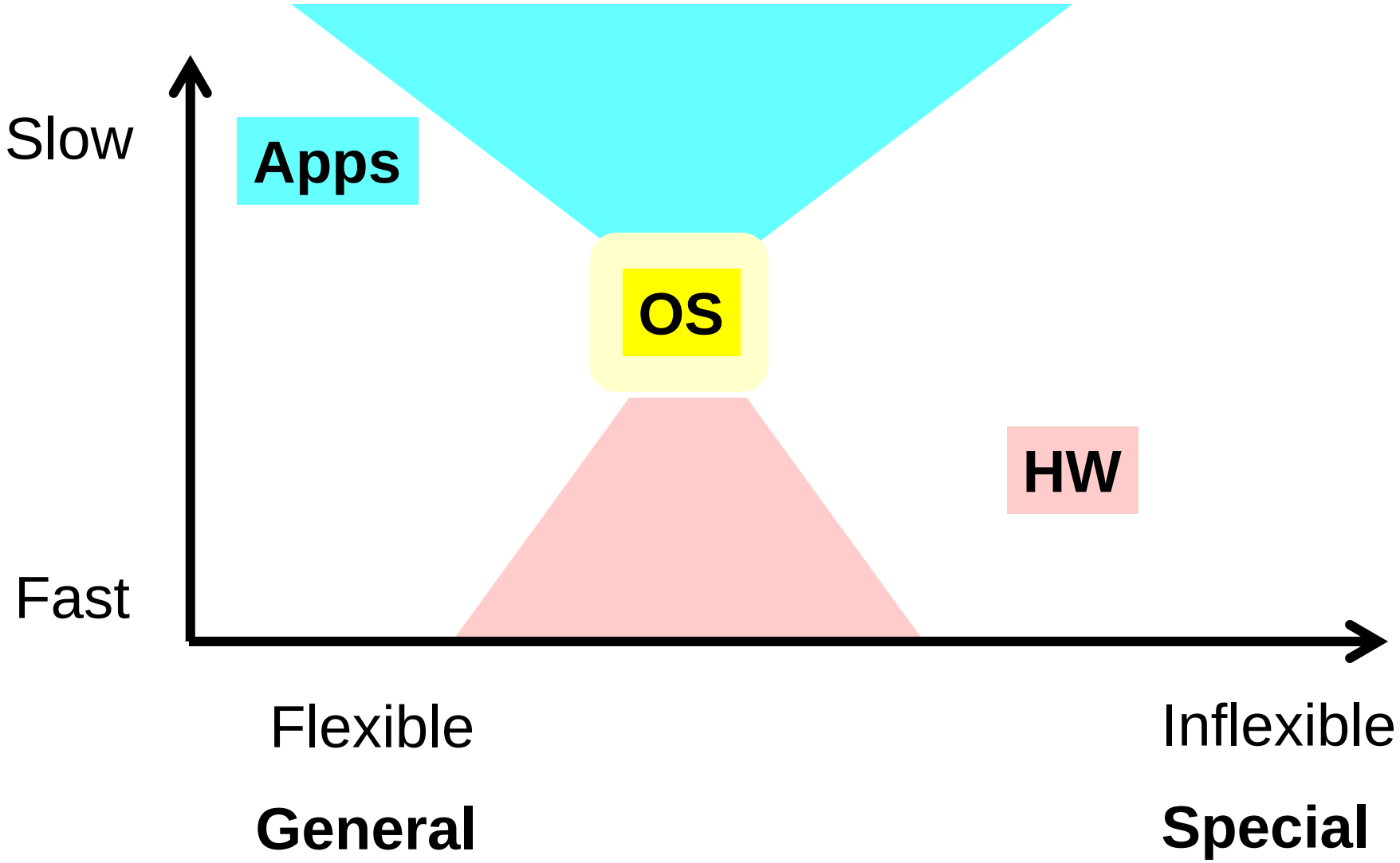
Apps

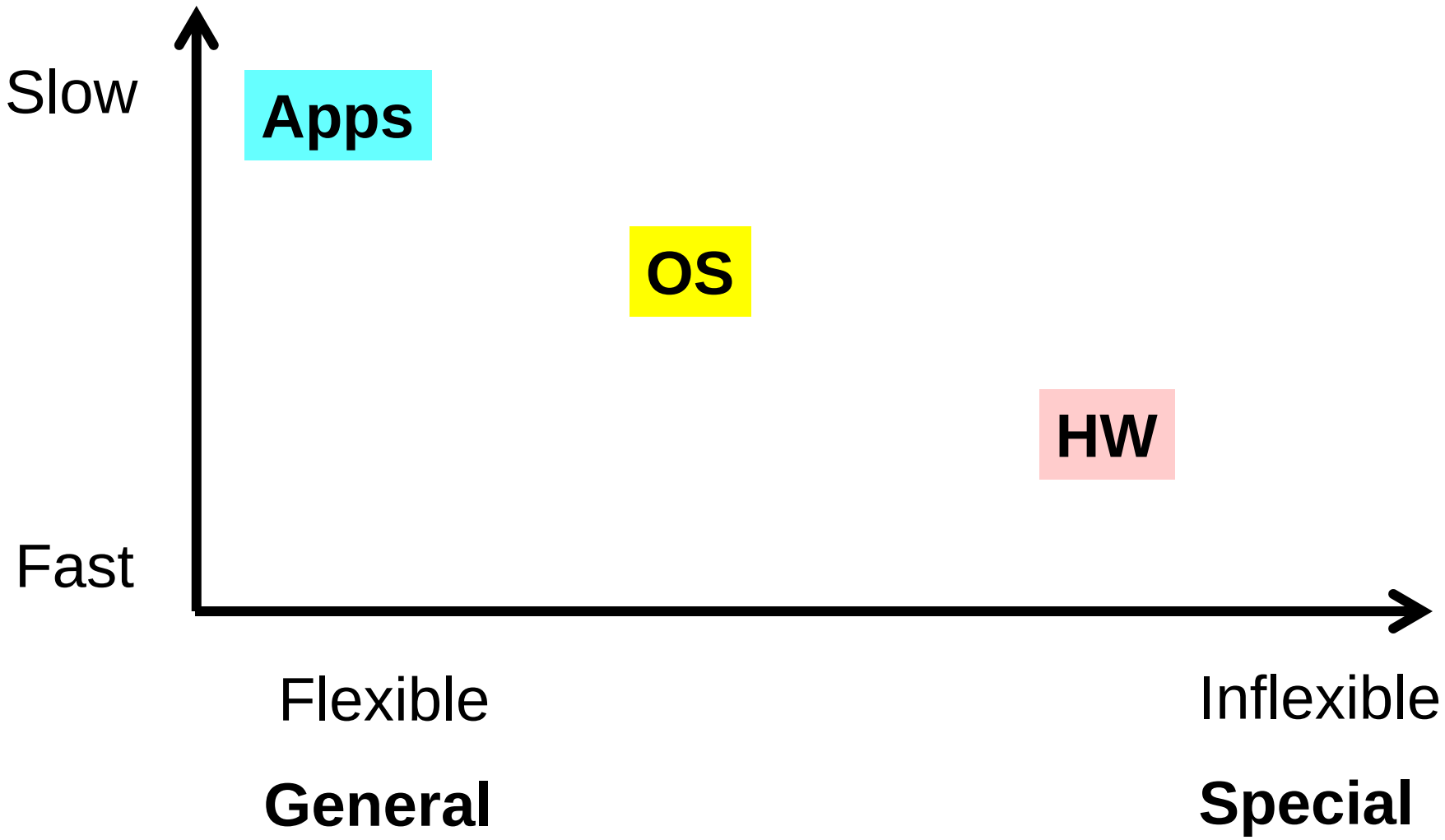
OS

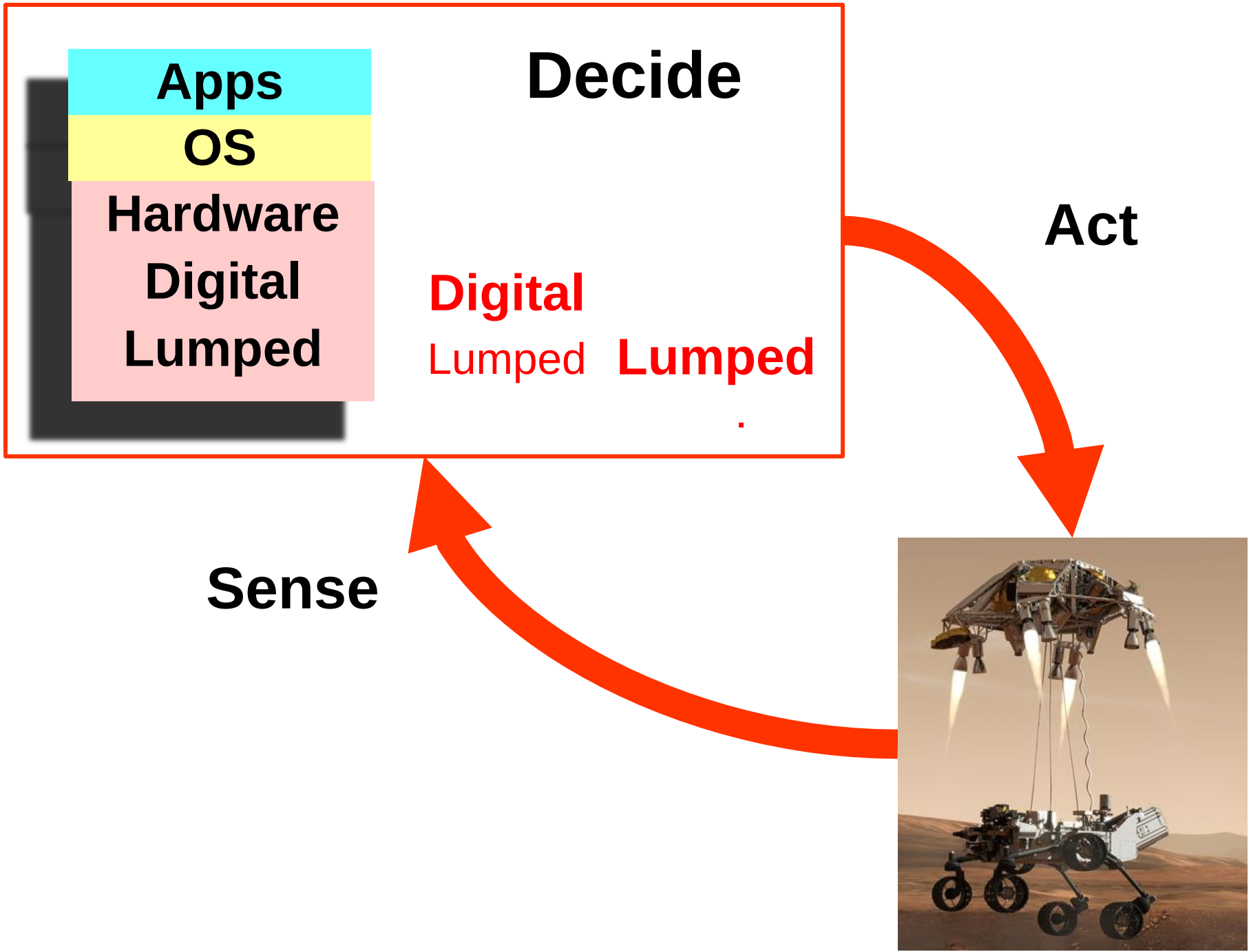
HW

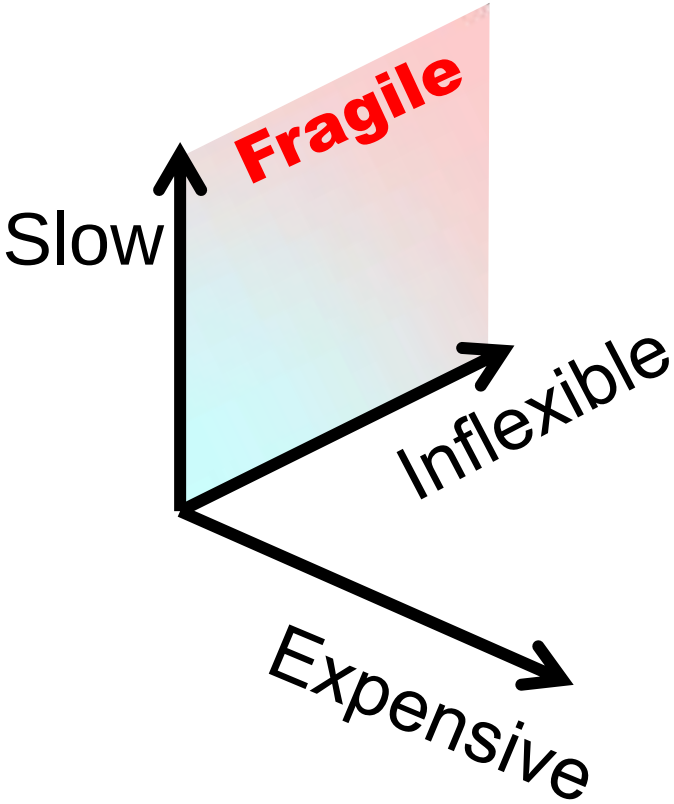
Layered architectures

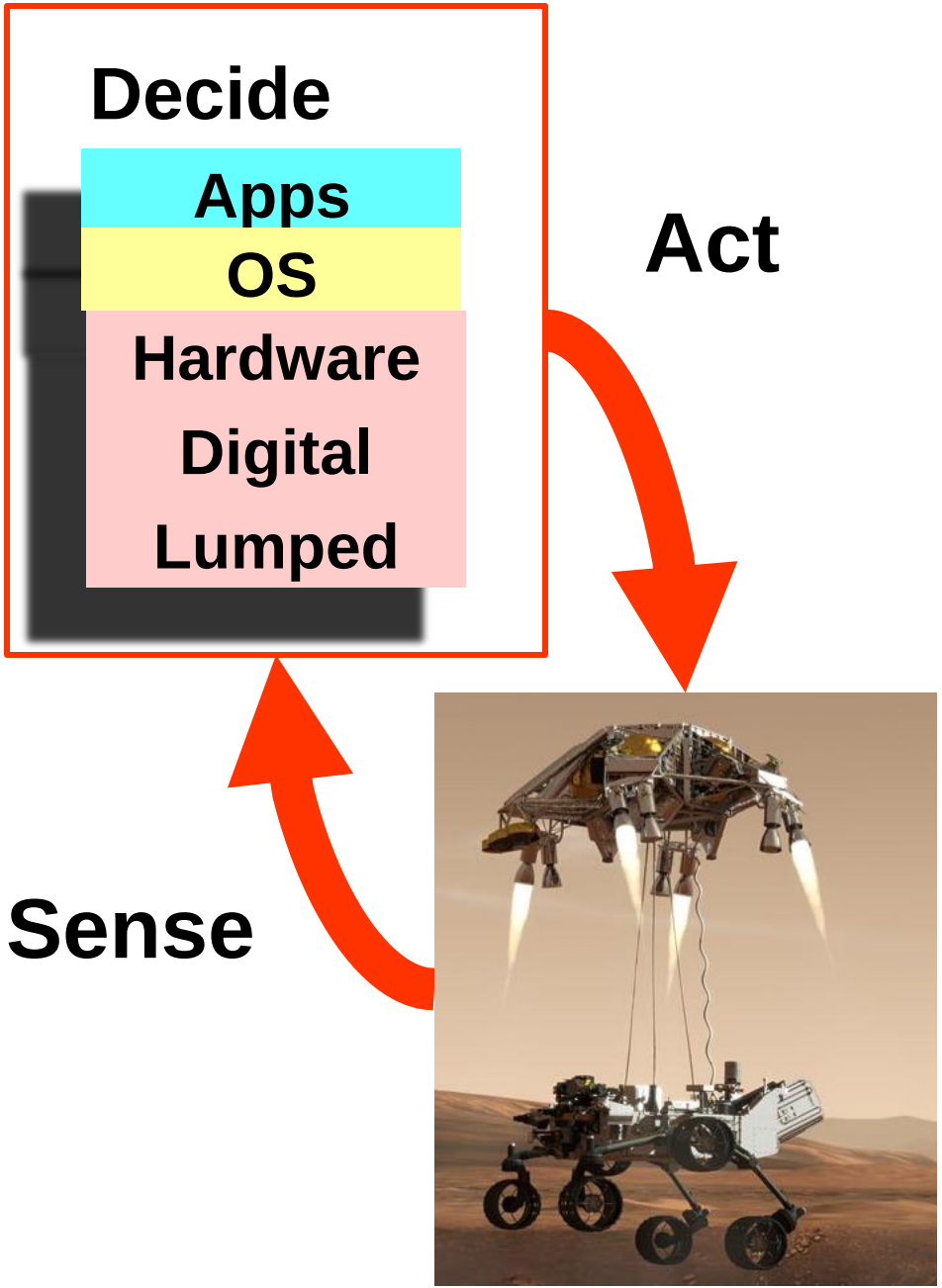



Slow
Apps
OS
HW
Fast
Flexible
Inflexible
General
Special


Slow
Apps
OS
HW
Fast
Flexible
Inflexible
General
Special

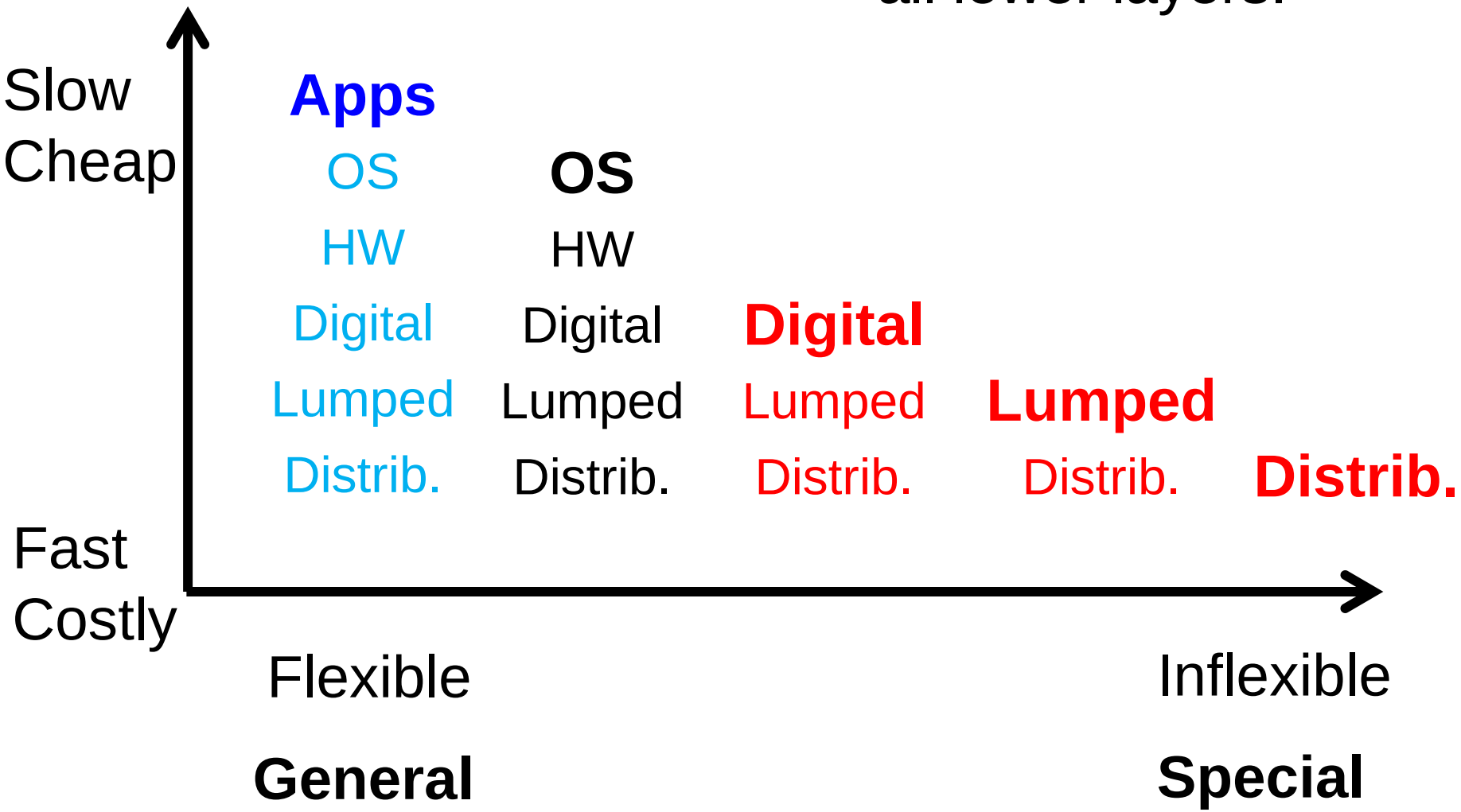

Apps
OS
Hardware
Digital
Lumped
Decide
Digital
Lumped
Lumped
Act
Sense


Fragile
Slow
Inflexible
Expensive

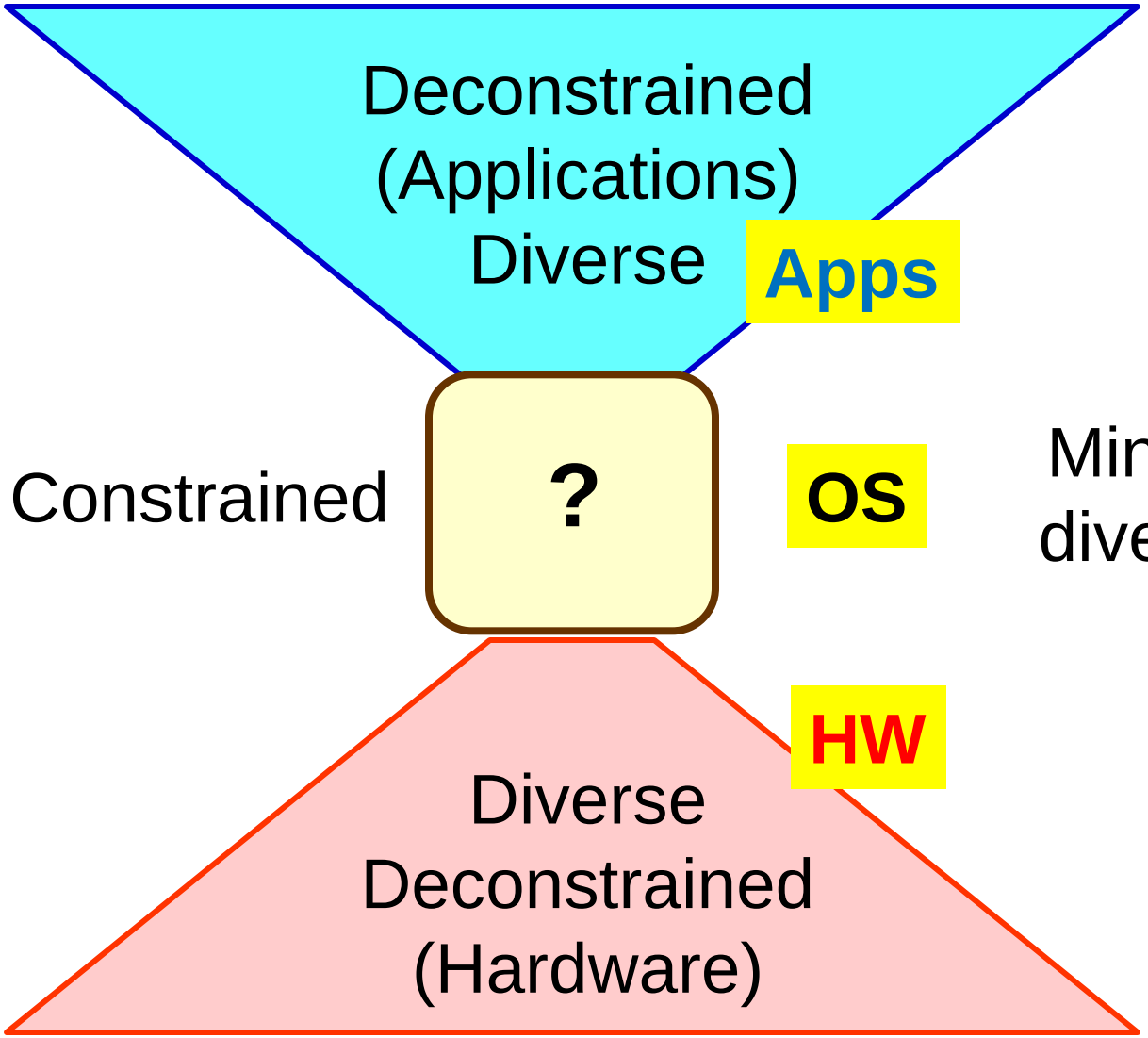

Decide
Apps
OS
Hardware
Digital
Lumped
Act
Sense

Layered Architecture

Any layer needs all lower layers.



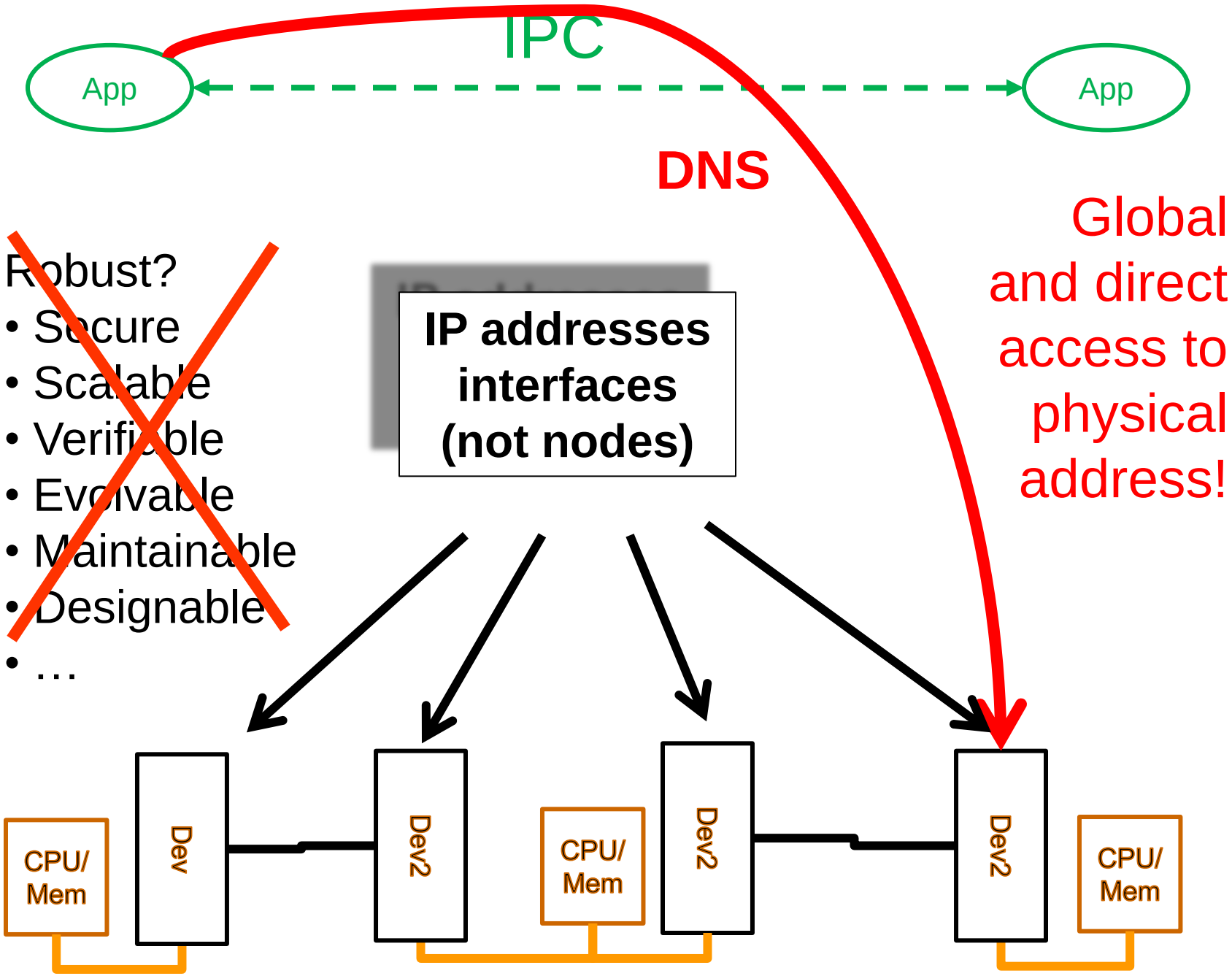
Layered architectures



Layered architectures make robustness and evolvability compatible

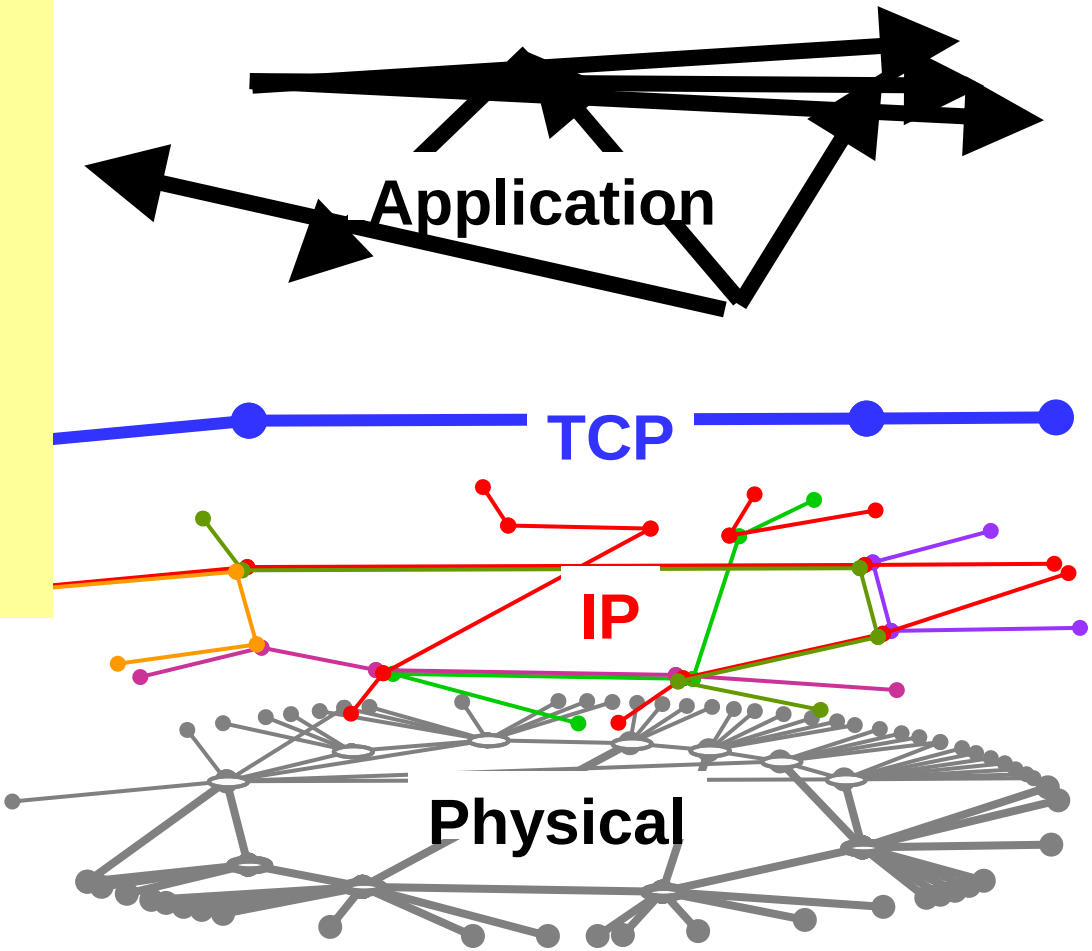
Minimal diversity

Ask me about hourglasses and bowties

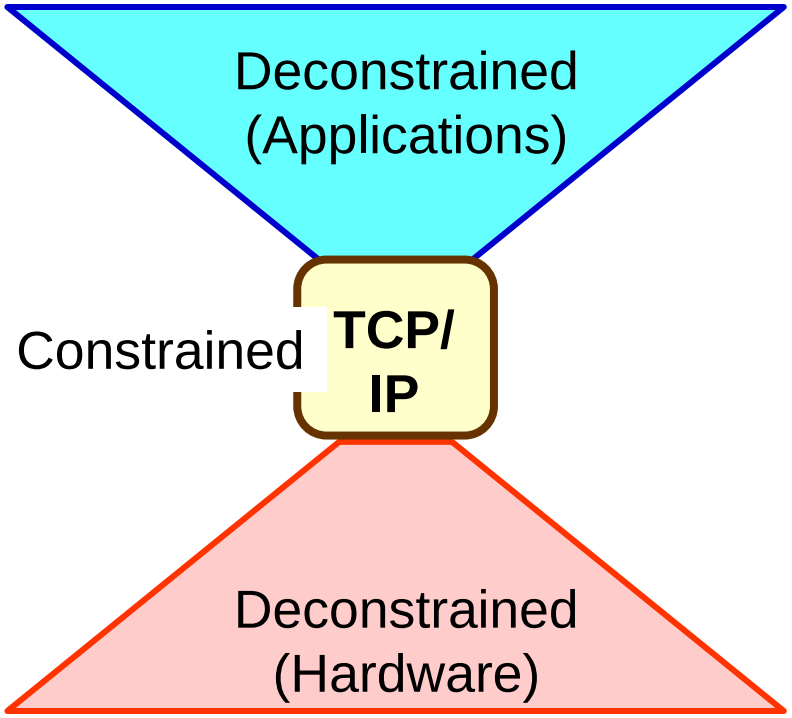

IPC
App
App
DNS
Global and direct access to physical address!
Robust?
• Secure
• Scalable
• Verifiable
• Evolvable
• Maintainable
• Designable
• …
IP addresses interfaces (not nodes)
CPU/ Mem
Dev
Dev2
CPU/ Mem
Dev2
Dev2
CPU/ Mem

“Issues” (hacks)

- VPNs
- NATS
- Firewalls
- Multihoming
- Mobility
- Routing table size
- Overlays
- …



Original design challenge?



Networked OS

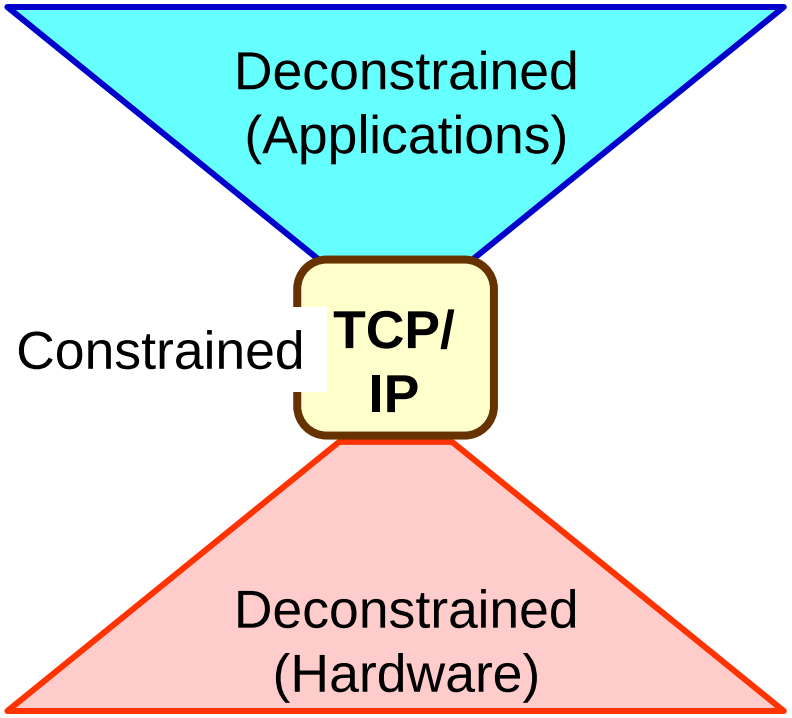
- ~~Expensive mainframes~~
- ~~Trusted end systems~~
- ~~Homogeneous~~
- ~~Sender centric~~
- ~~Unreliable comms~~

Facilitated wild evolution

Created

- whole new ecosystem
- completely opposite

Original design challenge?



Networked OS

Incomplete

- Layering (IP)
- Feedback (TCP)

- No theory

Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures

Starting point

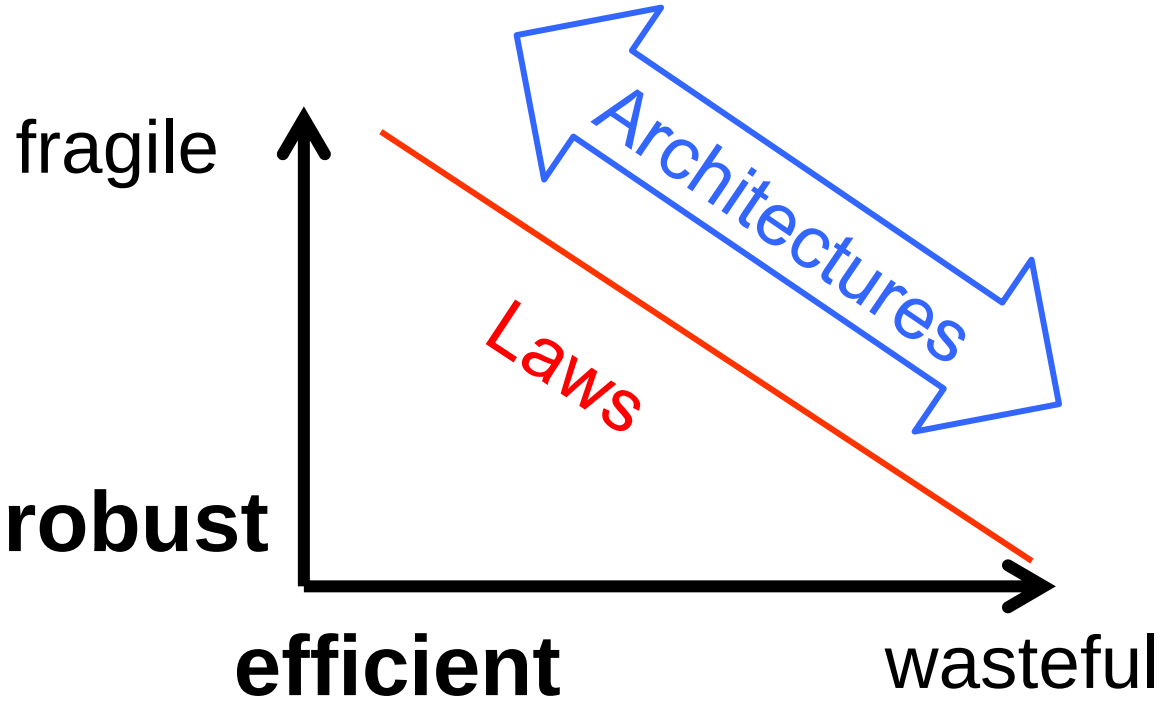
There are various ways that network functionalities can be allocated to different layers and to different network elements, some being more desirable than others. The intellectual goal of the research surveyed by this article is to provide a theoretical foundation for these architectural decisions in networking.

By Mung Chiang, *Member IEEE*, Steven H. Low, *Senior Member IEEE*, A. Robert Calderbank, *Fellow IEEE*, and John C. Doyle

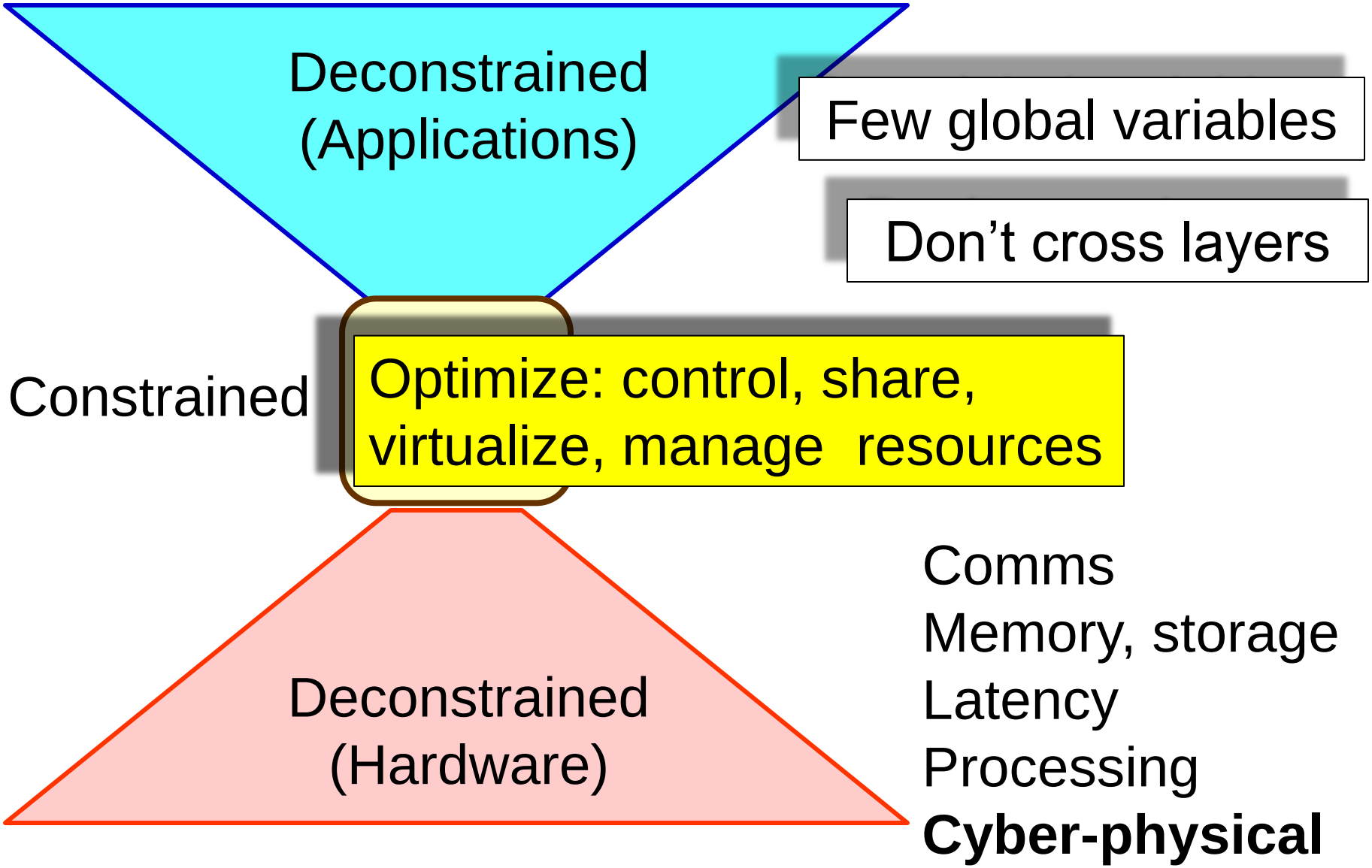
Chiang, Low, Calderbank, and Doyle

Vol. 95, No. 1, January 2007 | Proceedings of the IEEE

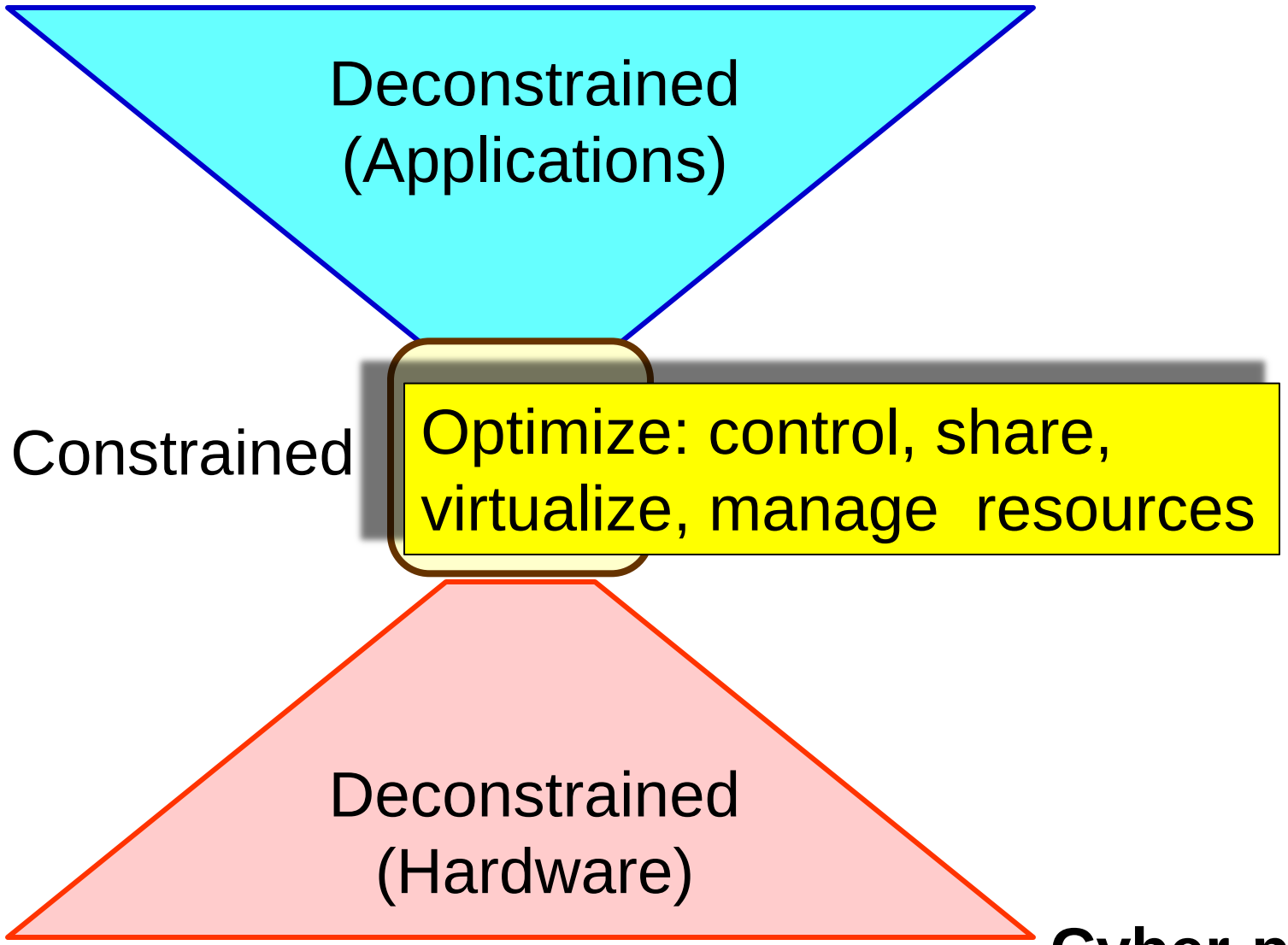
Layering
Feedback



Next layered arches (SDN? OpenDaylight?)

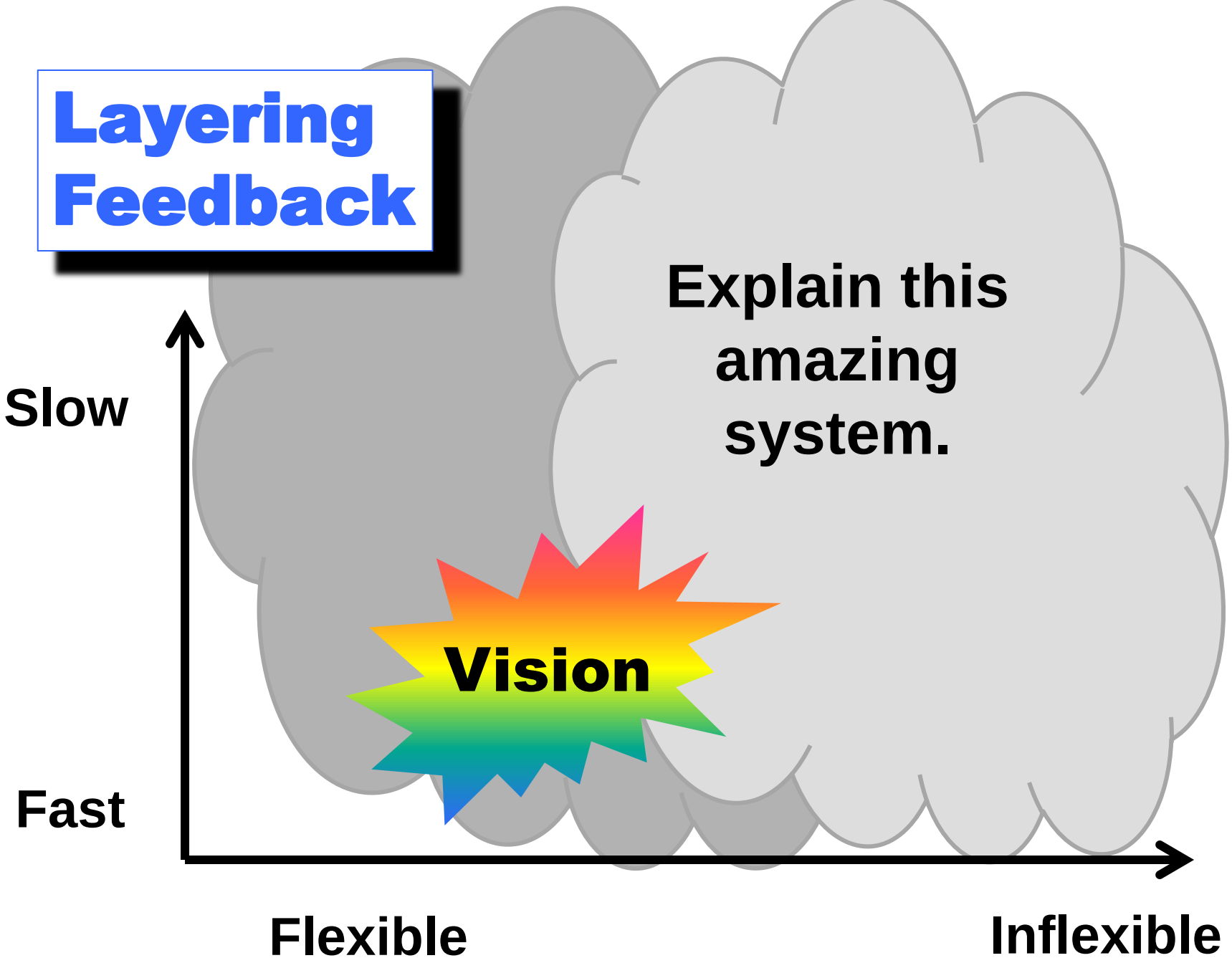


Next layered architectures



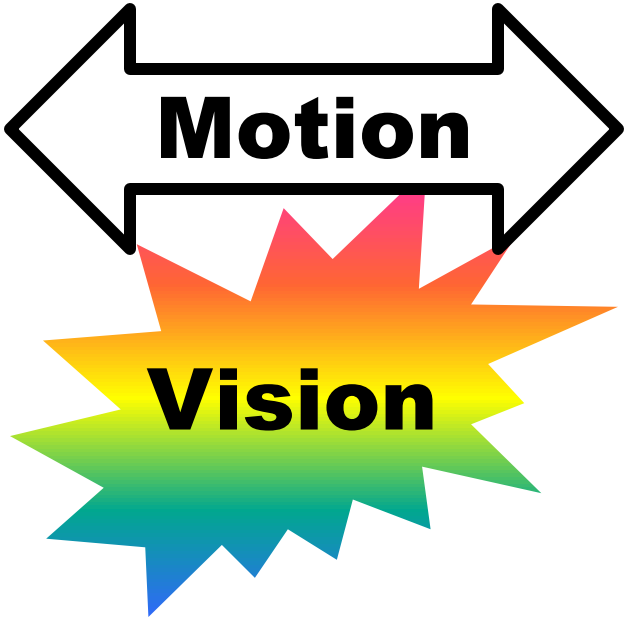
Cyber-physical

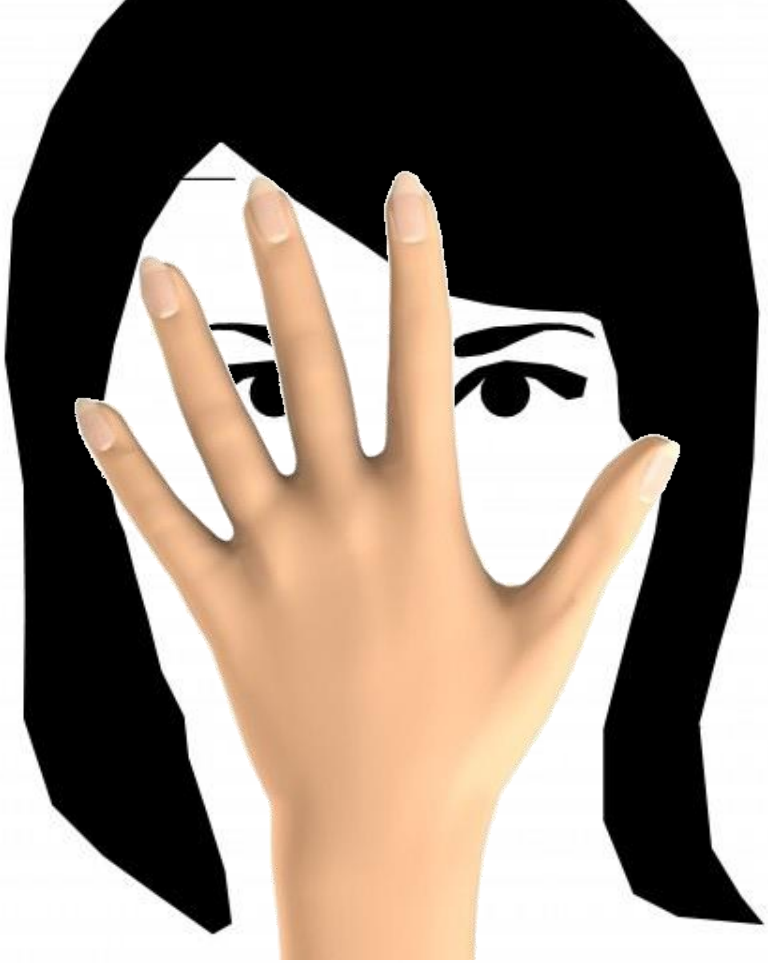
Layering Feedback



- **Neuroscience motivation**

Robust vision with motion





Experiment

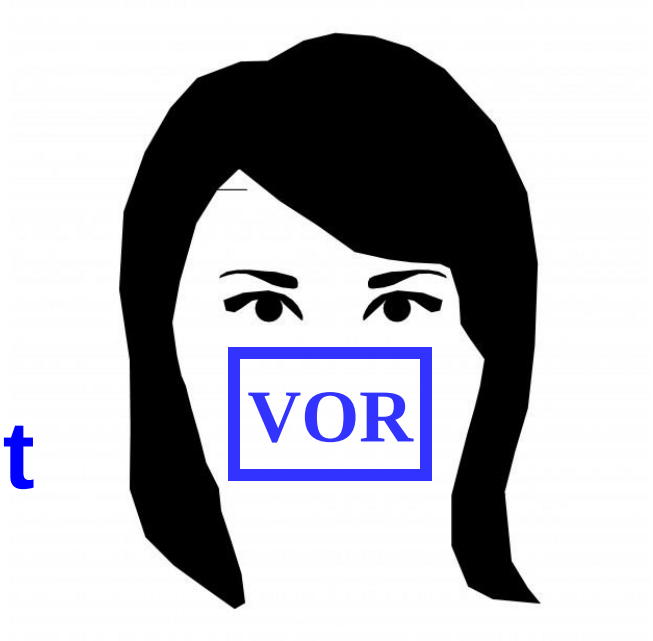
- Motion/vision control without blurring
- Which is easier?

Why?

- Mechanism
- Tradeoff

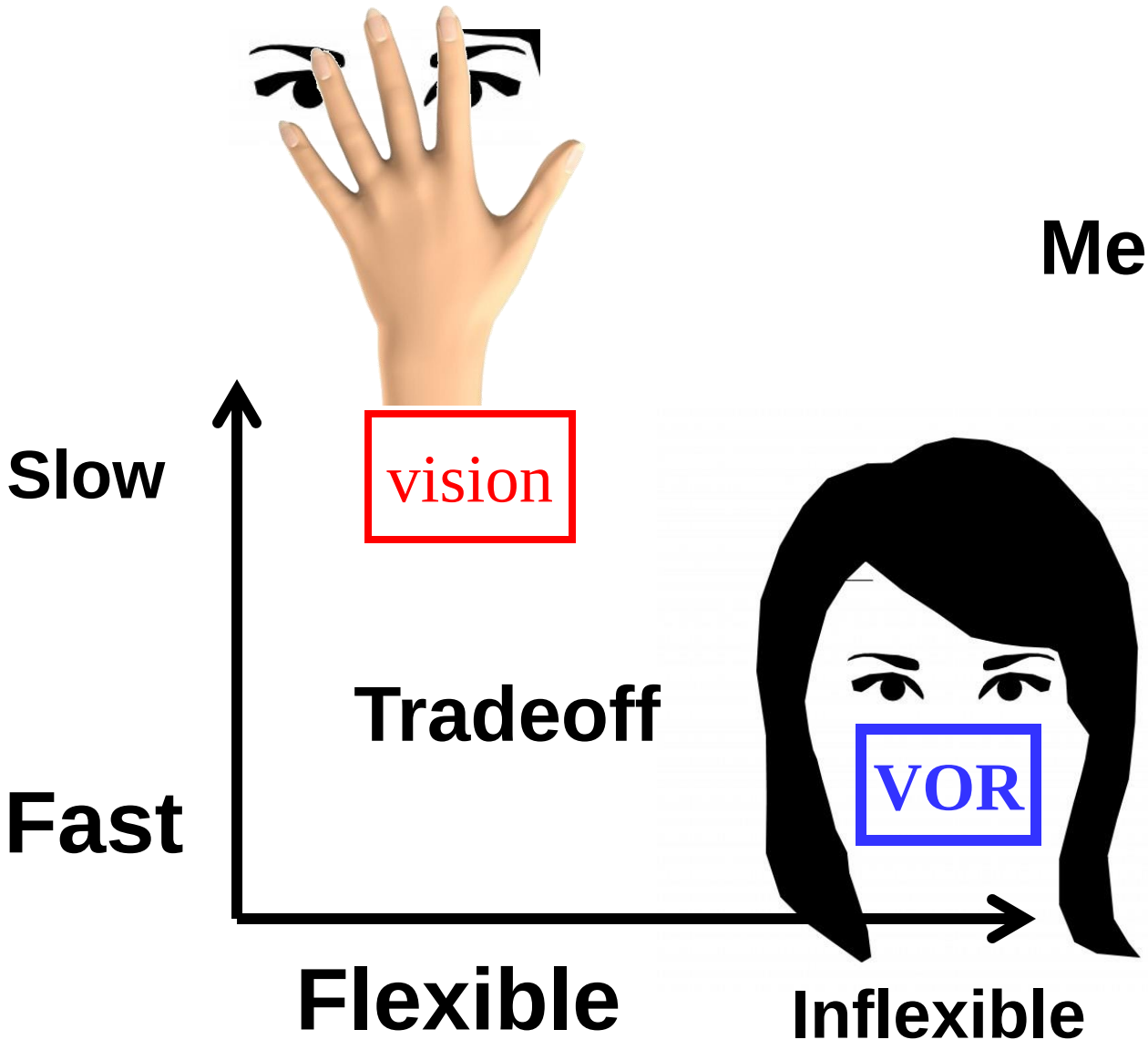
Slow

vision



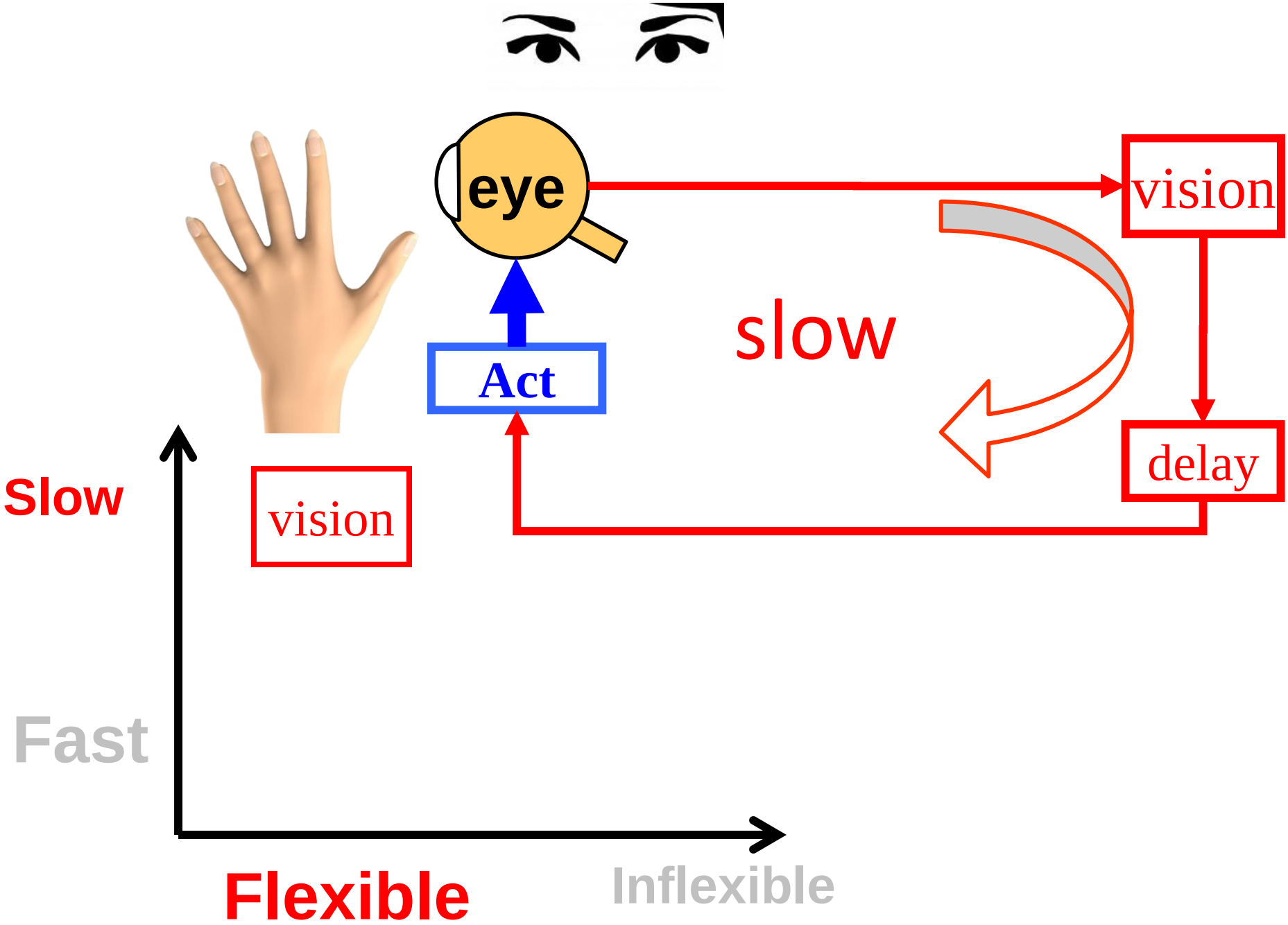
Fast

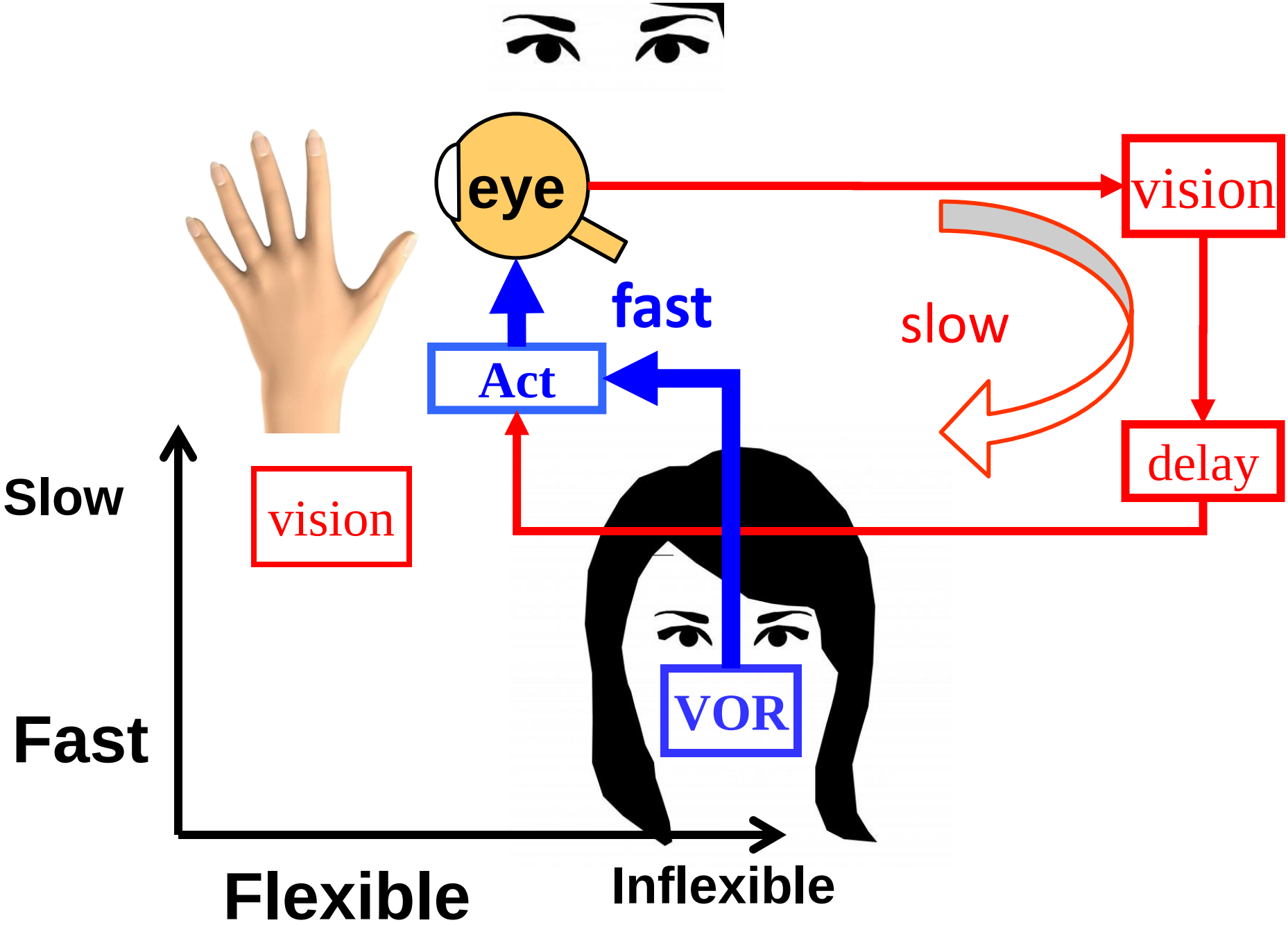
VOR

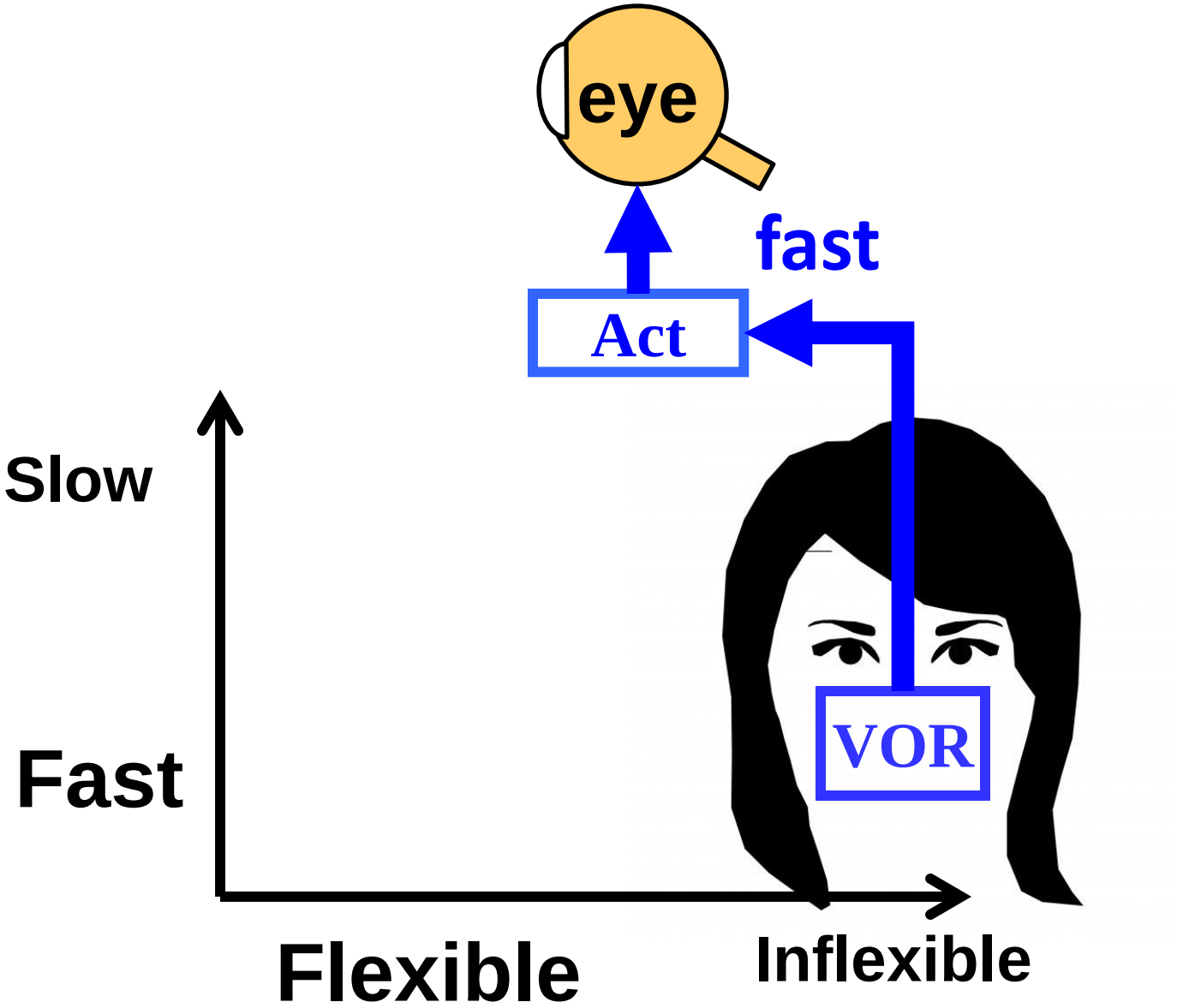


Mechanism

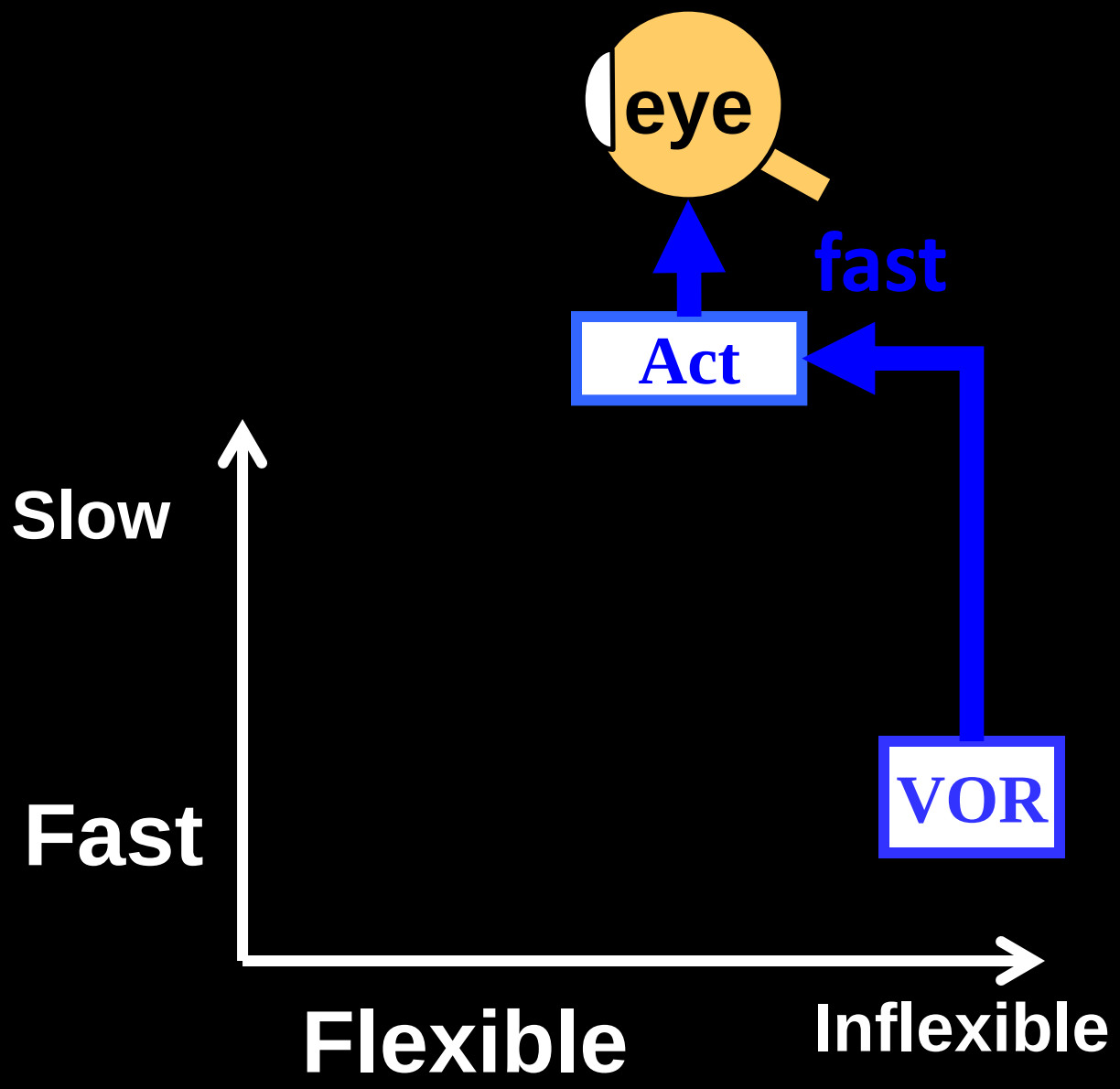
Vestibular Ocular Reflex (VOR)

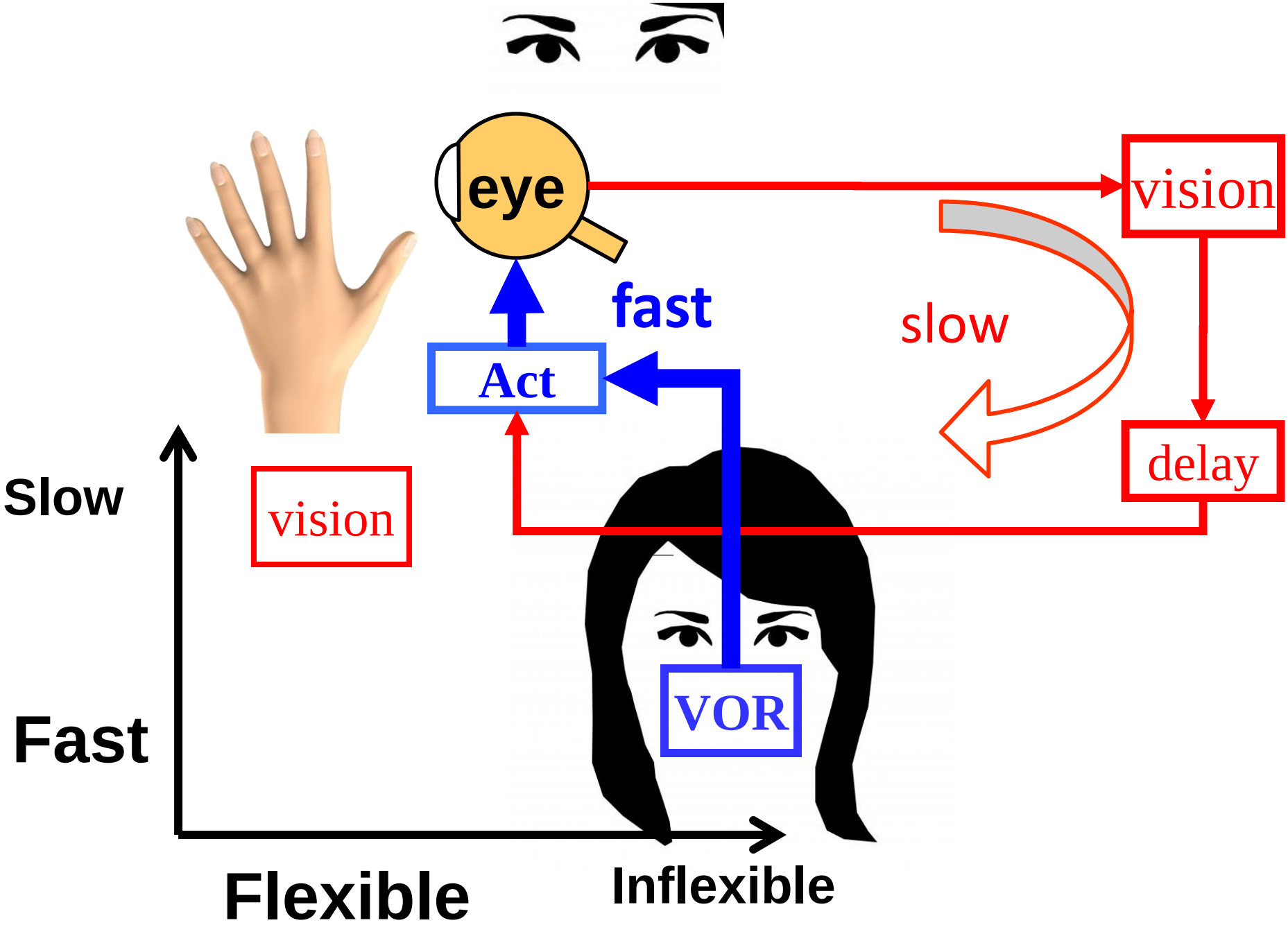

eye
vision
slow
Act
delay
vision
Slow
Fast
Flexible
Inflexible

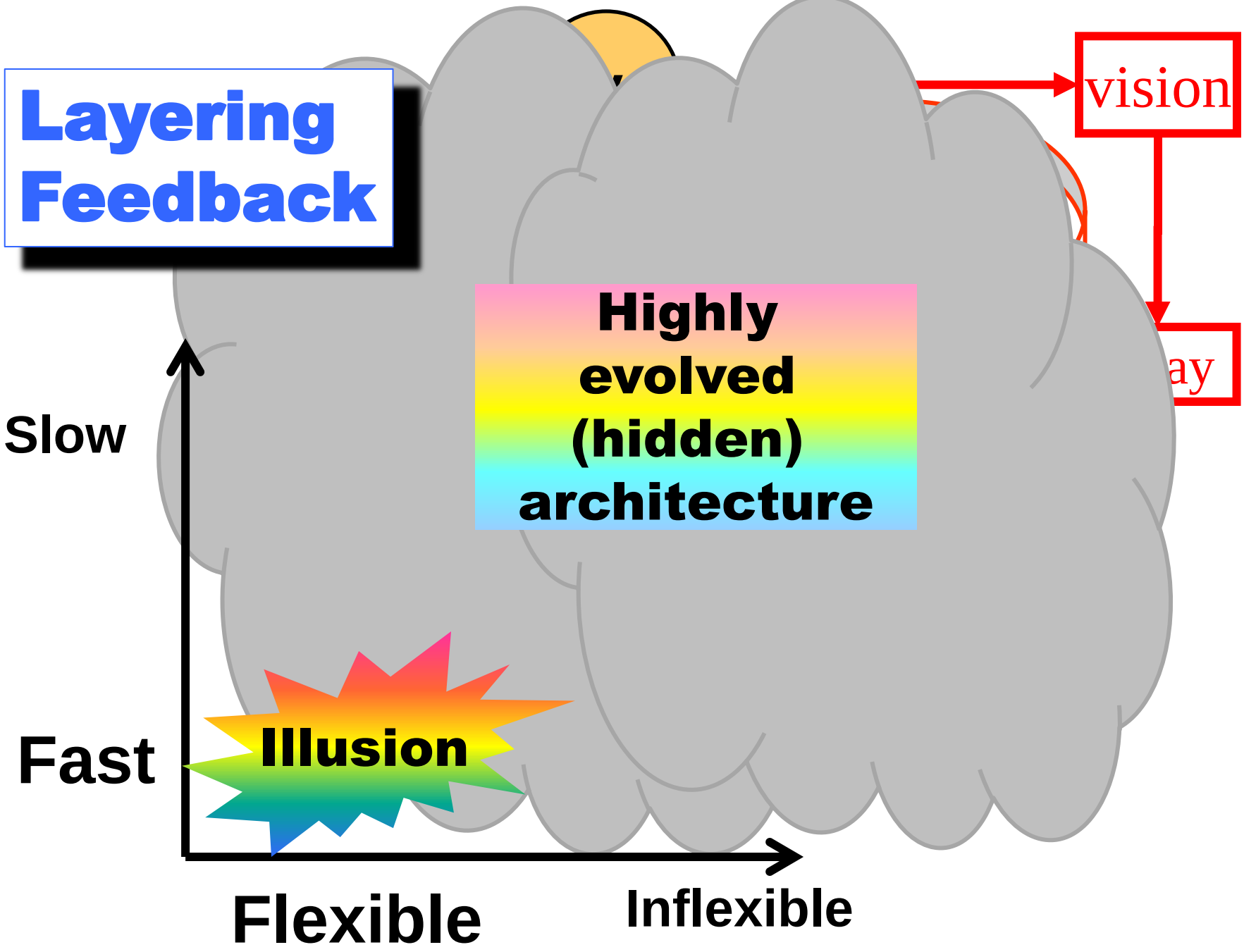

eye
vision
fast
slow
Act
delay
vision
Slow
Fast
VOR
Flexible
Inflexible



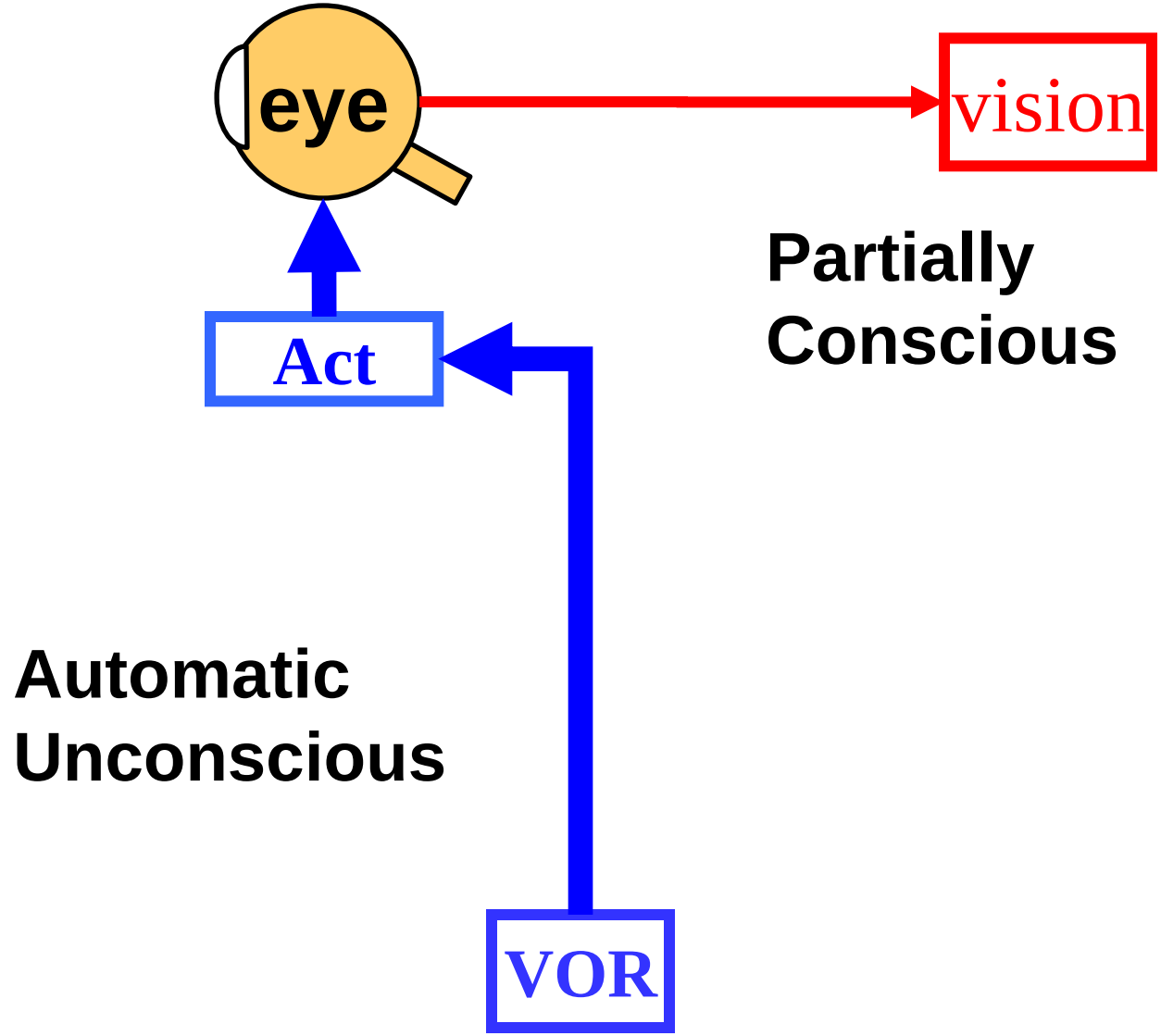
Vestibular Ocular Reflex (VOR)


eye
fast
Act
VOR
Slow
Fast
Flexible
Inflexible

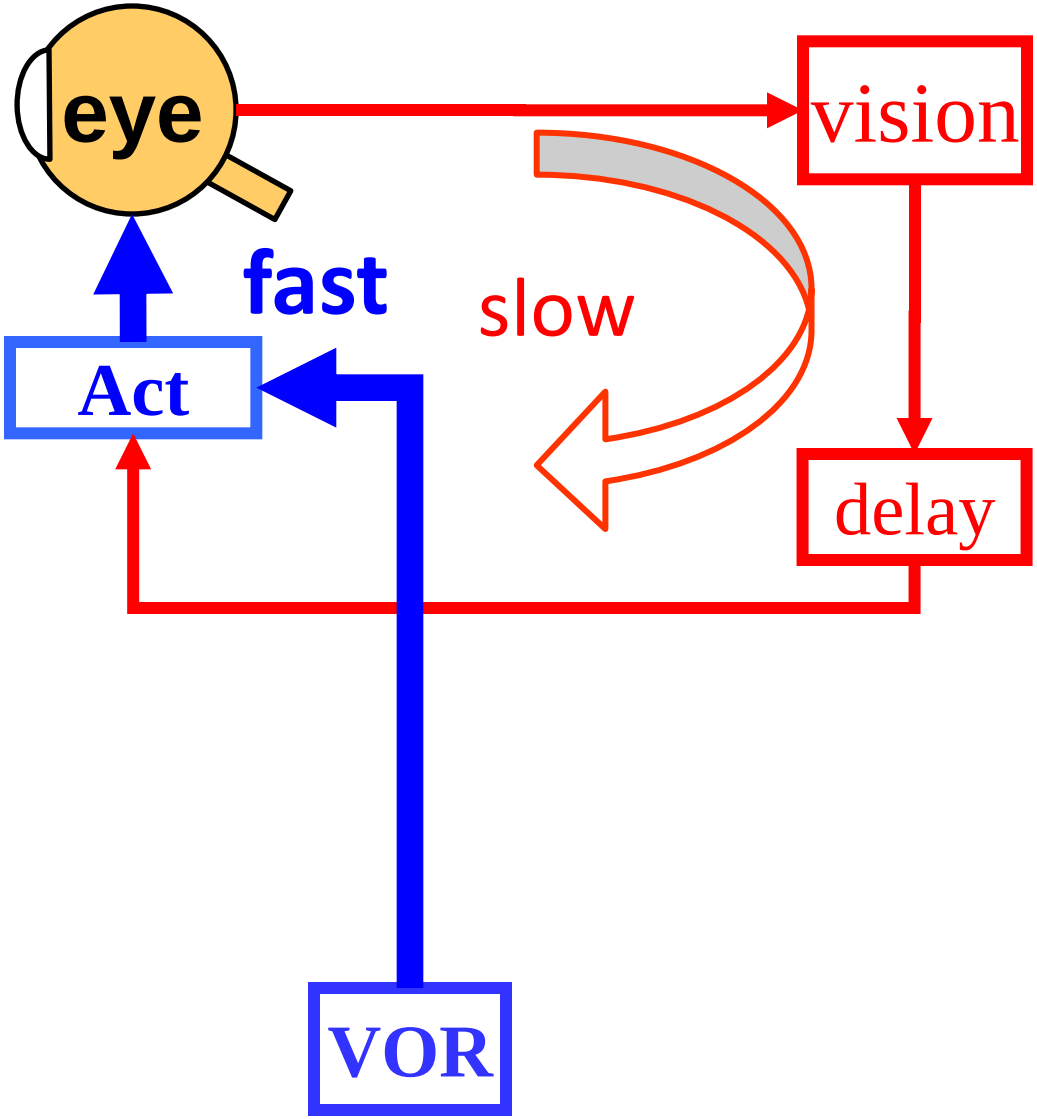

eye
vision
fast
slow
Act
delay
vision
Slow
Fast
VOR
Flexible
Inflexible


Layering
Feedback
vision
ay
Highly
evolved
(hidden)
architecture
Slow
Fast
Illusion
Flexible
Inflexible

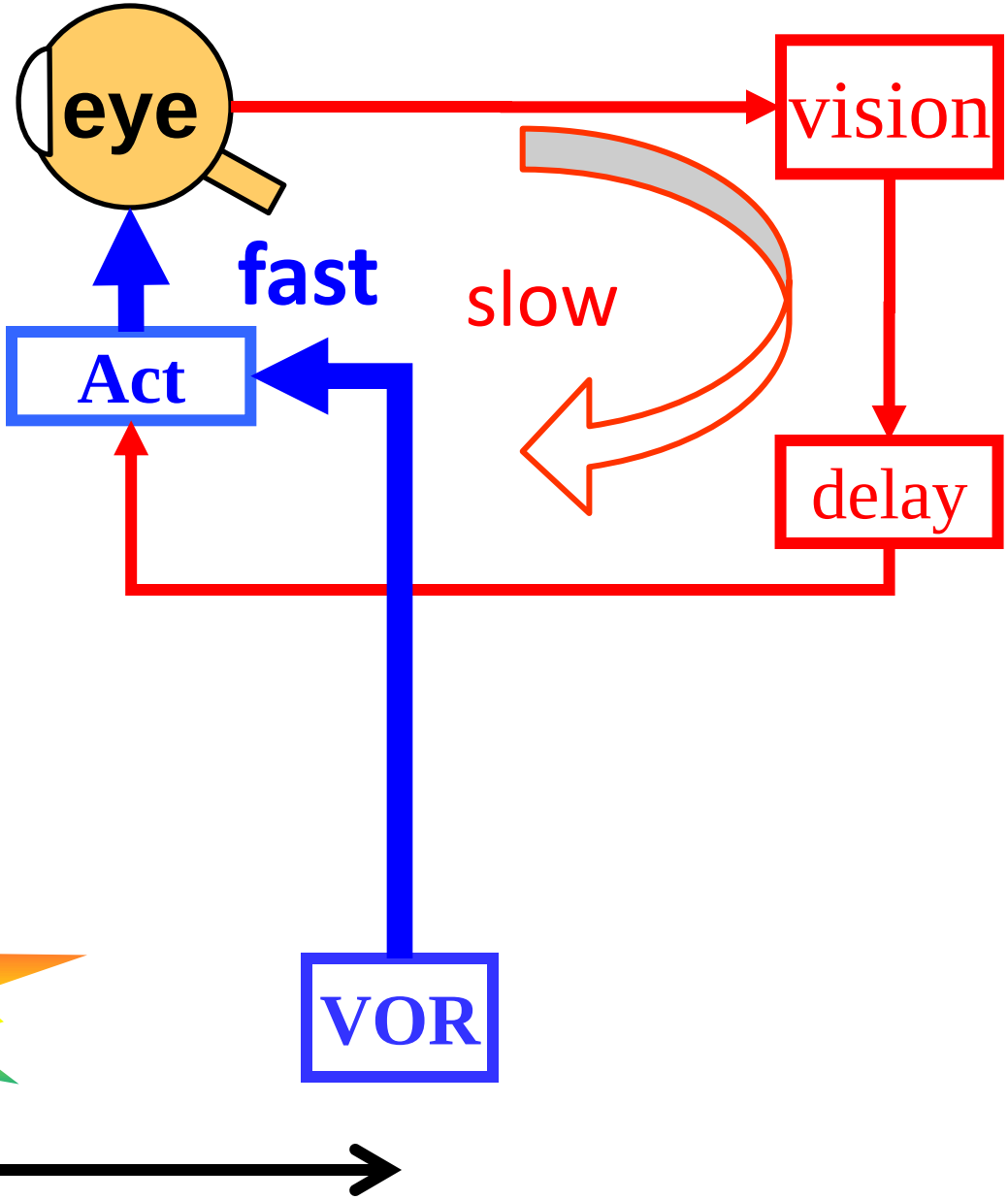
Layering

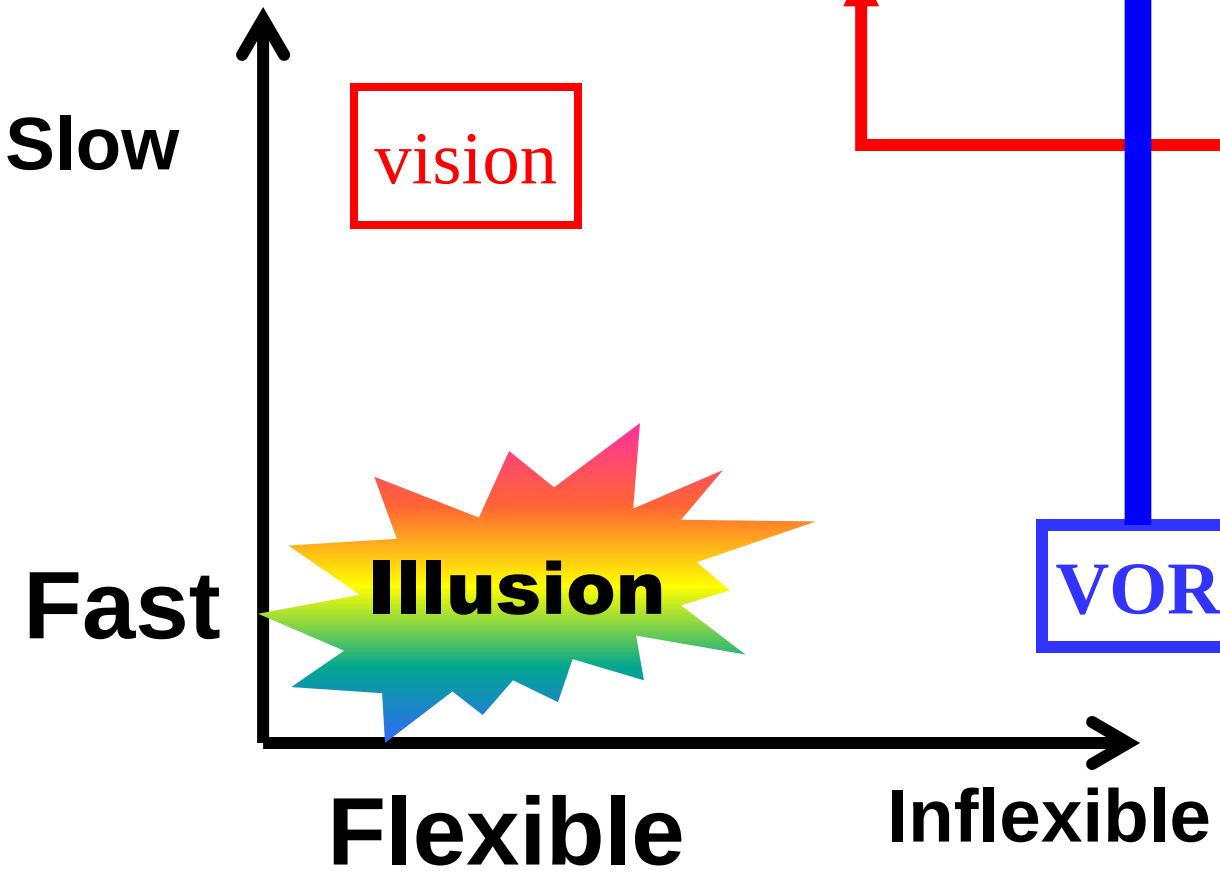

eye
vision
Partially
Conscious
Act
Automatic
Unconscious
VOR

Layering Feedback



Architecture

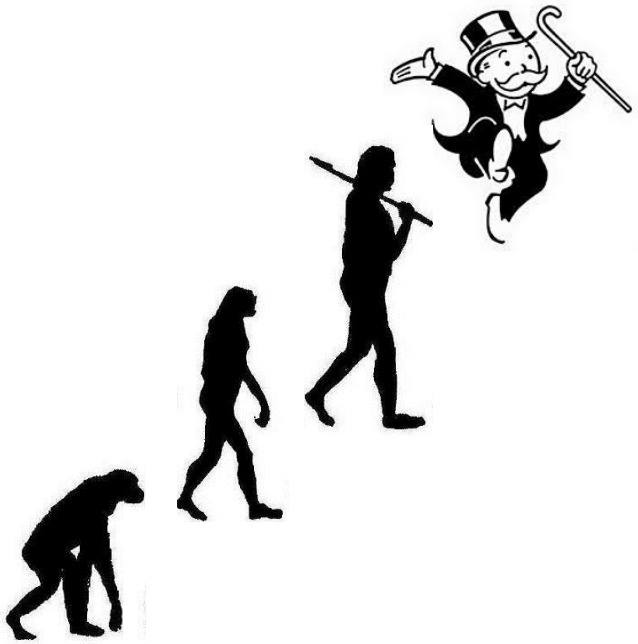




Layers/feedback in biology

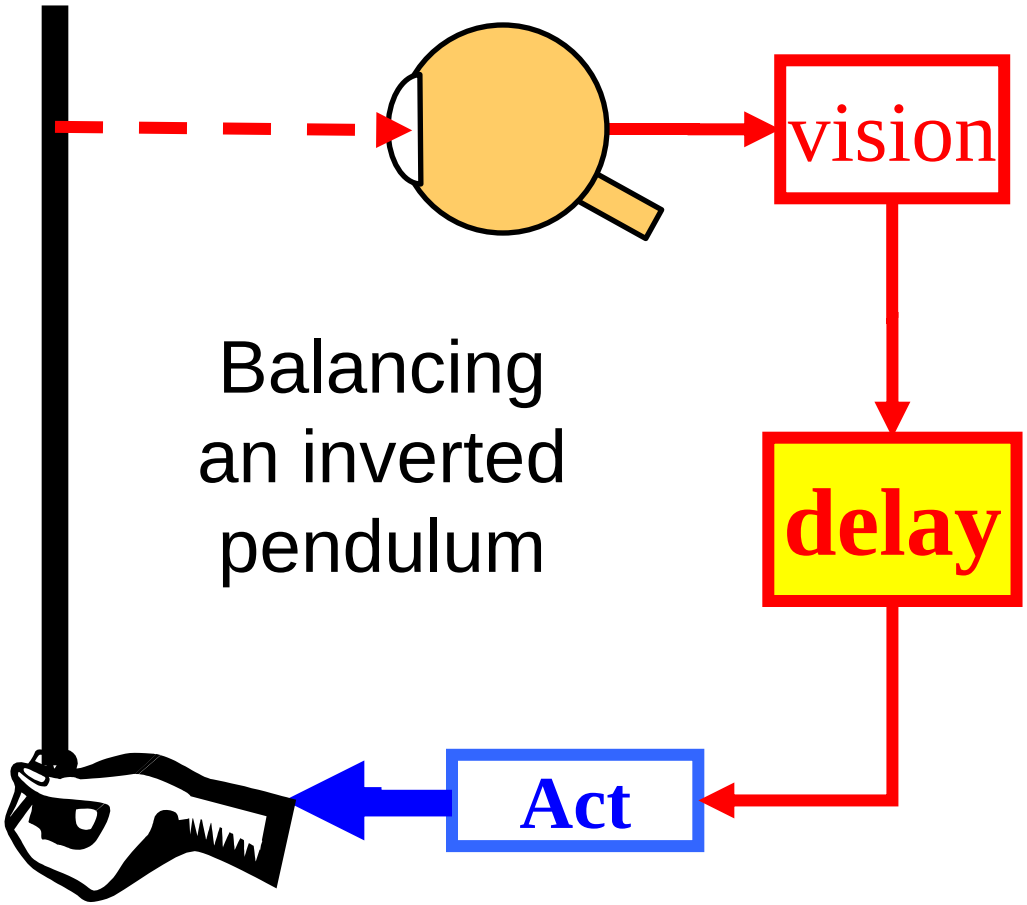
Working backwards

- Money/finance/lobbyists/etc
- Society/agriculture/weapons/etc
- Bipedalism
- Maternal care
- Warm blood
- Flight
- Mitochondria
- Translation (ribosomes)
- Glycolysis (see 2011 Science paper)



Universal laws

Mechanics+
Gravity +
Light +



Balancing
an inverted
pendulum

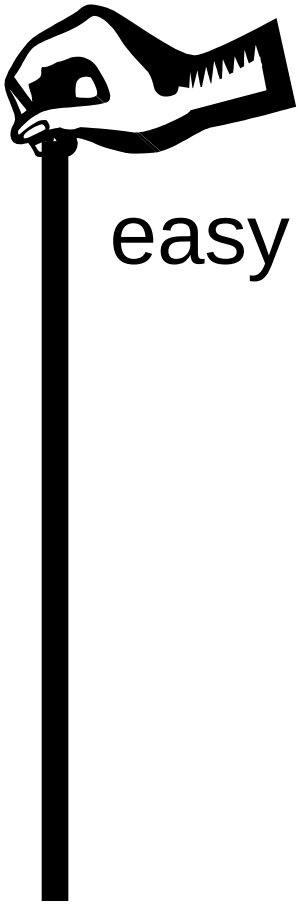
$$\frac{1}{\pi}\int_0^\infty \ln\left|T(j\omega)\right|\left(\frac{p}{p^2+\omega^2}\right)d\omega$$

$$\geq p\tau + \ln\left|\frac{z+p}{z-p}\right|$$

+ Neuroscience

Law #1 : Mechanics

Law #2 : Gravity



easy

$$(M+m)\ddot{x} + ml\left(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right) = u$$

$$\ddot{x}\cos\theta + l\ddot{\theta} + g\sin\theta = 0$$

$$y = x + \alpha l\sin\theta$$

linearize

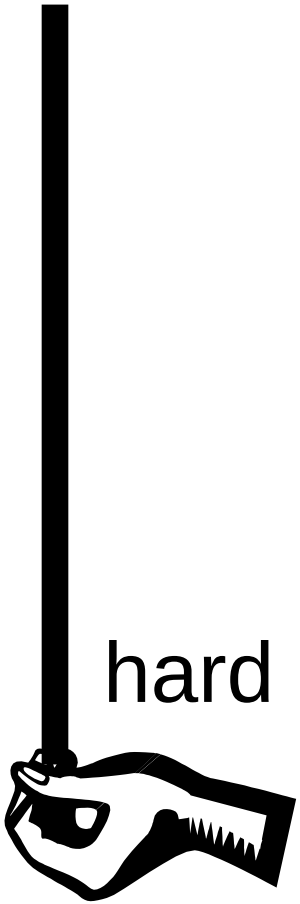
$$(M+m)\ddot{x} + ml\ddot{\theta} = u$$

$$\ddot{x} + l\ddot{\theta} + g\theta = 0$$

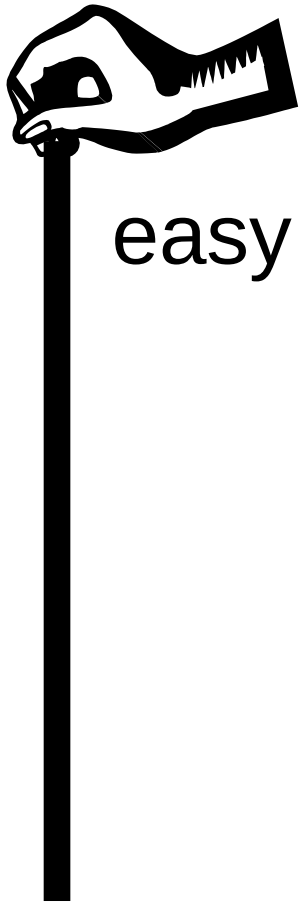
$$y = x + \alpha l\theta$$

Law #1 : Mechanics

Law #2 : Gravity



hard



easy

$$(M+m)\ddot{x} + ml\left(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right) = u$$

$$\ddot{x}\cos\theta + l\ddot{\theta} + g\sin\theta = 0$$

$$y = x + \alpha l\sin\theta$$

linearize

$$(M+m)\ddot{x} + ml\ddot{\theta} = u$$

$$\ddot{x} + l\ddot{\theta} + g\theta = 0$$

$$y = x + \alpha l\theta$$

Crashes can be made rare with active control.

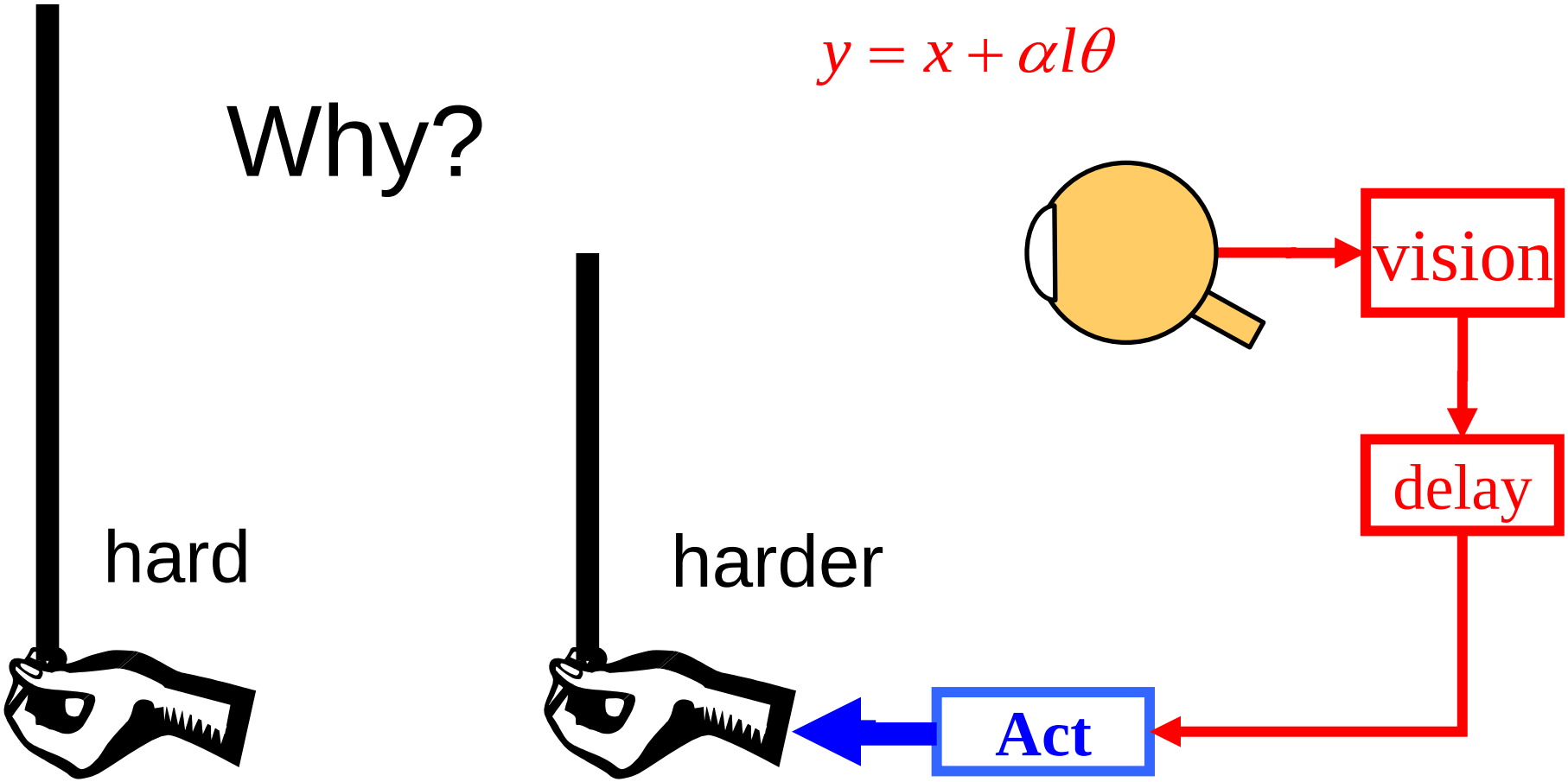
Law #3 : Light

$$(M+m)\ddot{x} + ml\ddot{\theta} = u$$

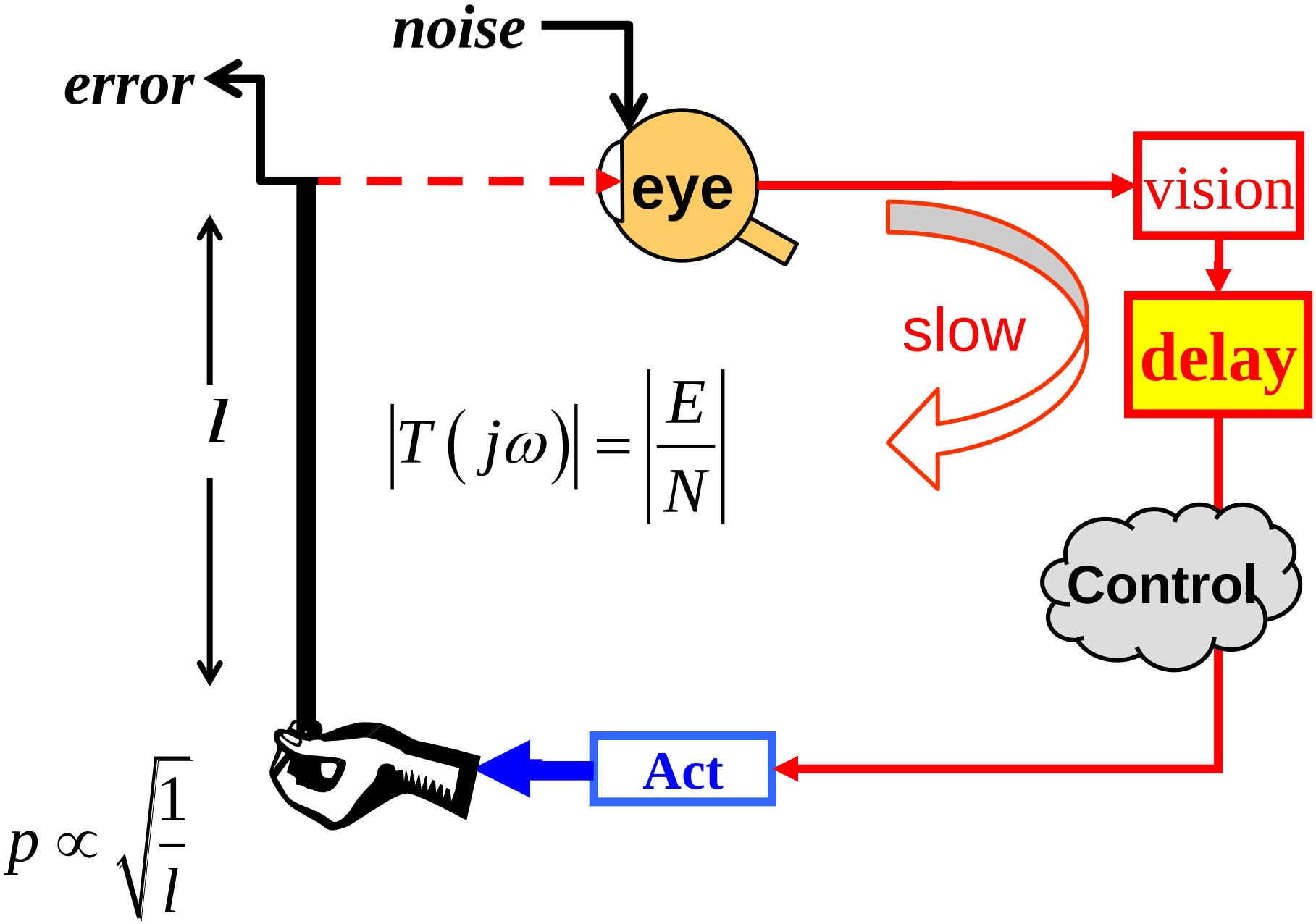
$$\ddot{x} + l\ddot{\theta} \pm g\theta = 0$$

$$y = x + \alpha l\theta$$

Why?

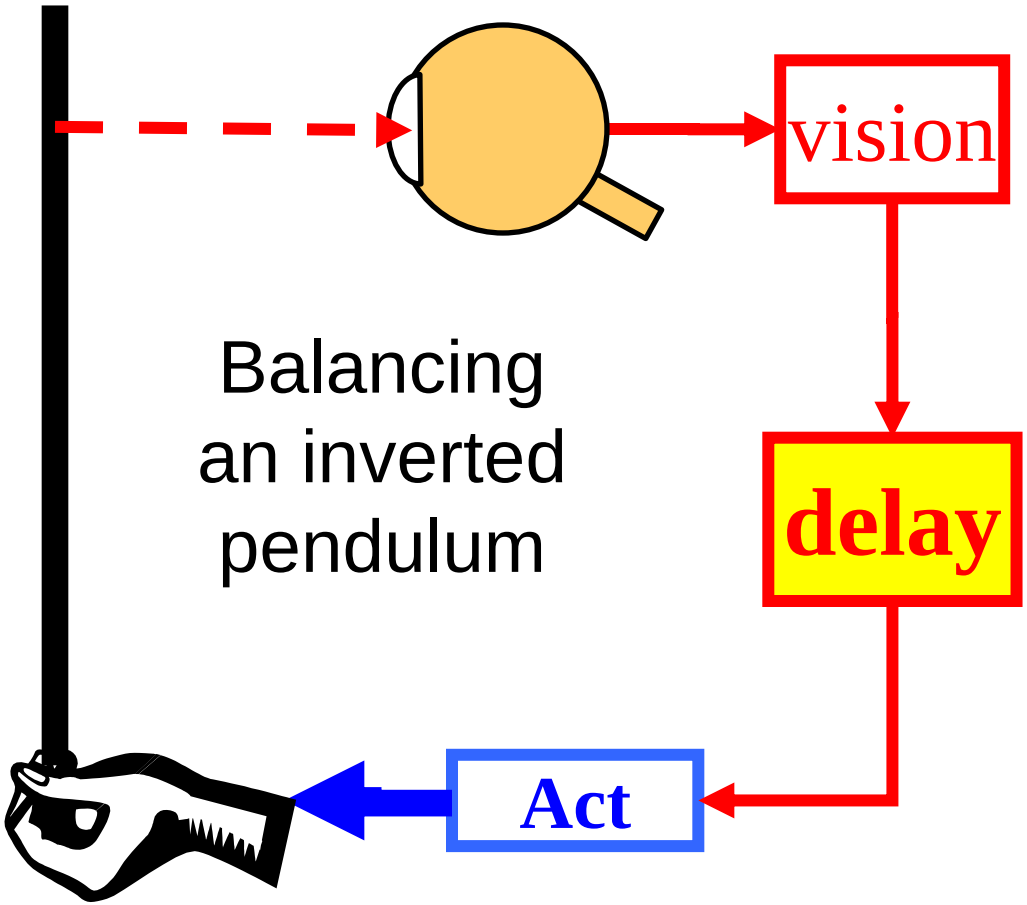


Easy to ***prove*** using simple models.


noise
error
eye
vision
slow
delay
l
$|T(j\omega)| = \left|\frac{E}{N}\right|$
Control
Act
$p \propto \sqrt{\frac{1}{l}}$

Universal laws

Mechanics+
Gravity +
Light +



Balancing
an inverted
pendulum

$$\frac{1}{\pi}\int_0^\infty \ln\left|T(j\omega)\right|\left(\frac{p}{p^2+\omega^2}\right)d\omega$$

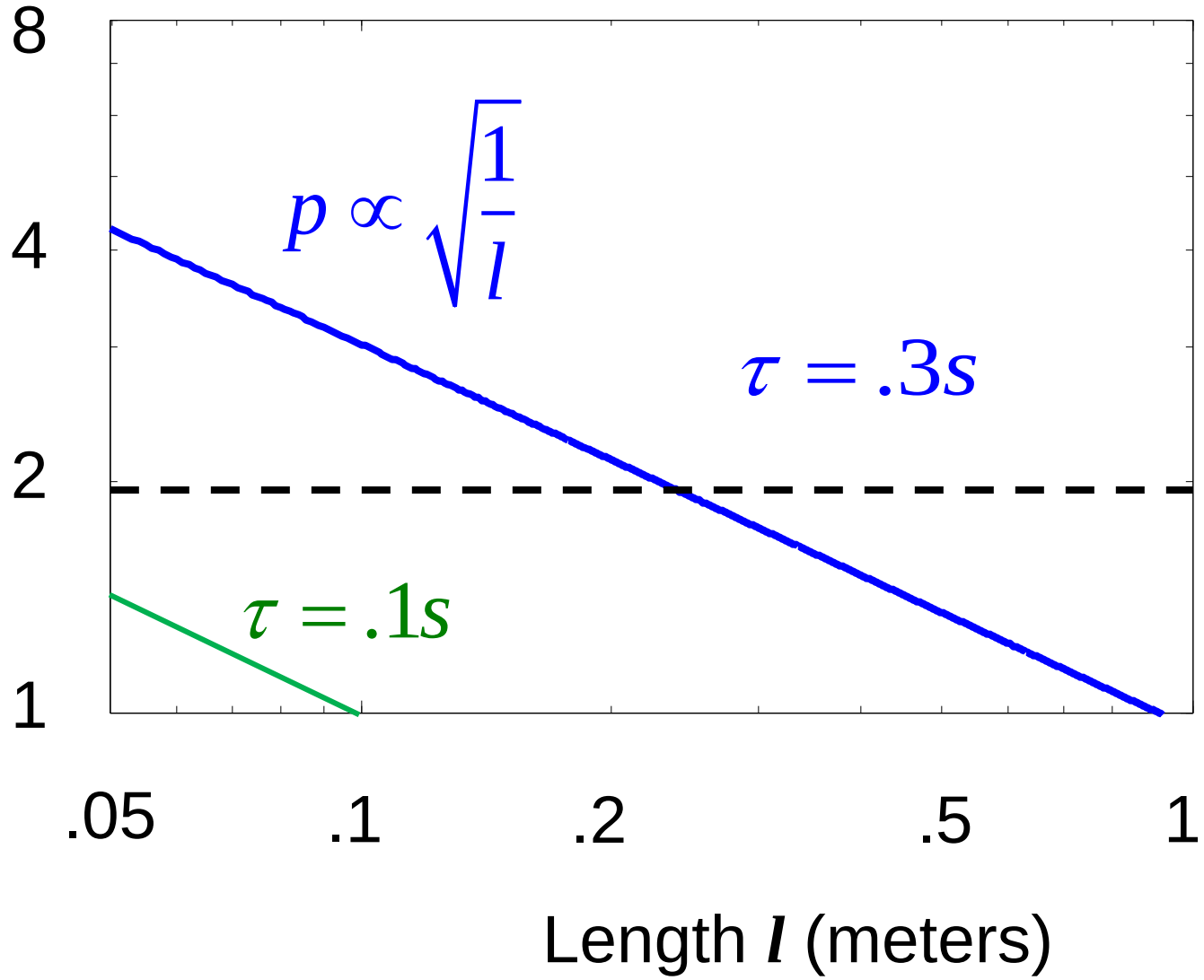
$$\geq p\tau + \ln\left|\frac{z+p}{z-p}\right|$$

+ Neuroscience

Law #4 : $$\frac{1}{\pi}\int_0^\infty \ln\left|T(j\omega)\right|\left(\frac{2p}{p^2+\omega^2}\right)d\omega \geq p\tau$$

Fragility

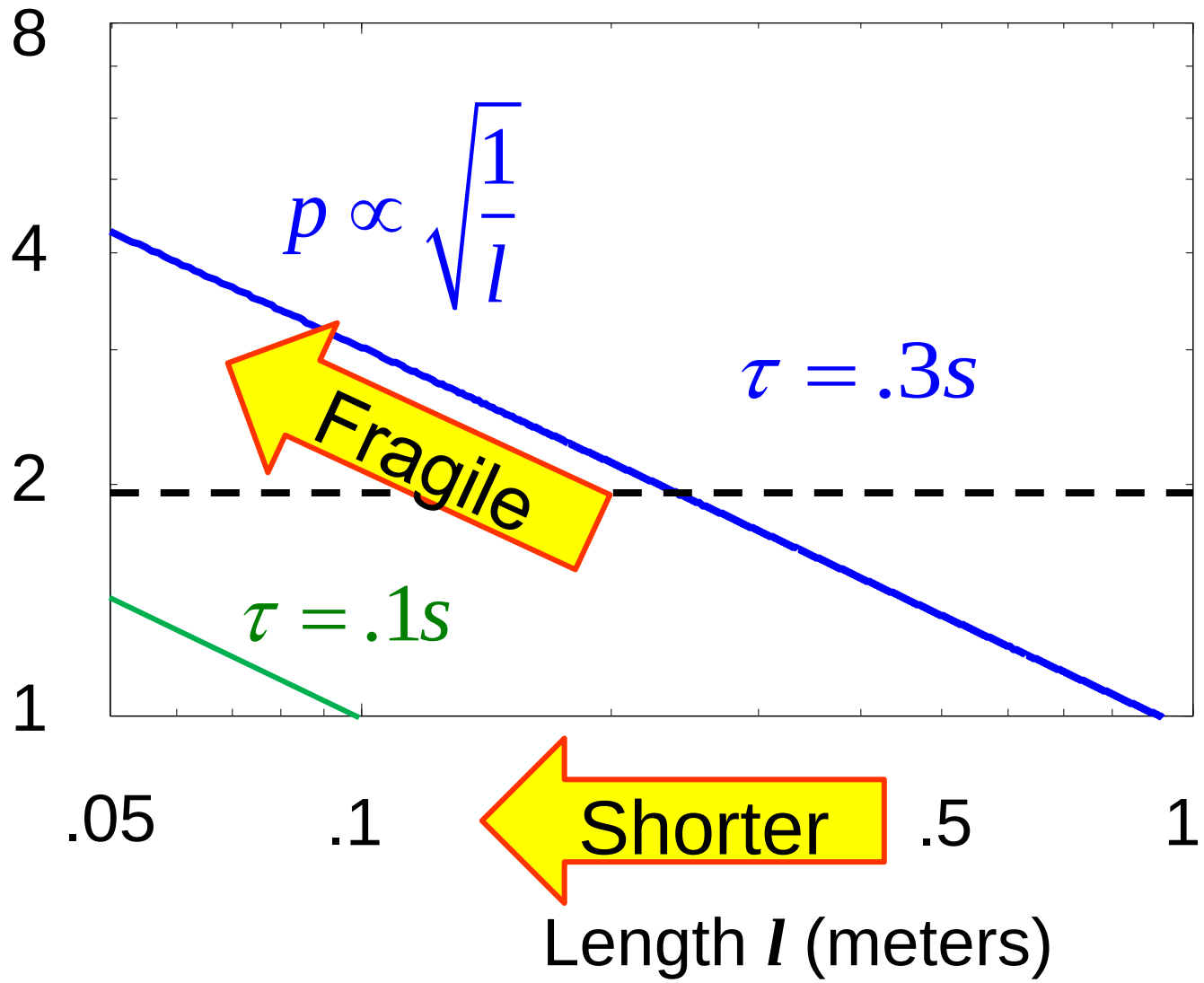
$p\tau$



Law #4 : $$\frac{1}{\pi}\int_0^\infty \ln\left|T(j\omega)\right|\left(\frac{2p}{p^2+\omega^2}\right)d\omega \geq p\tau$$

Fragility

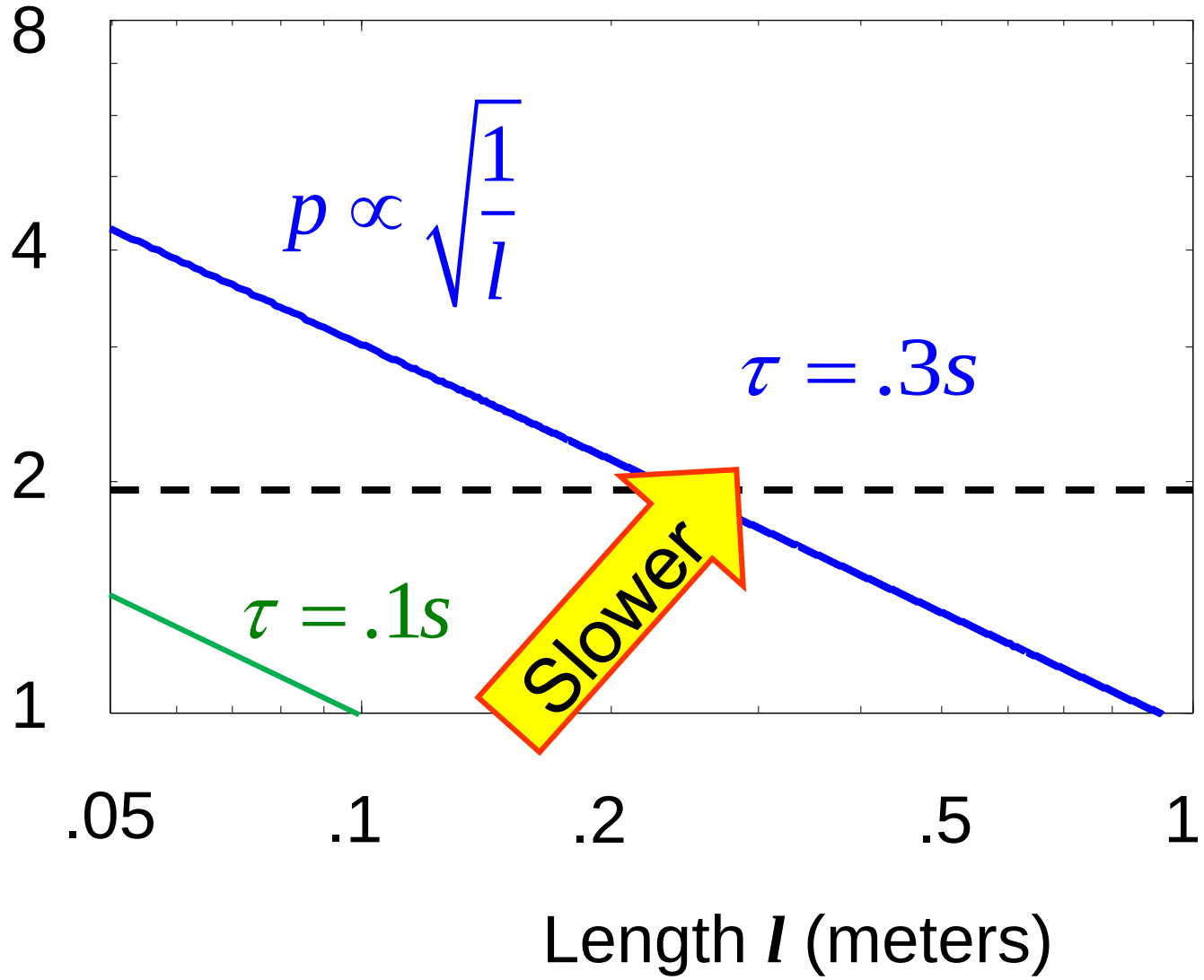
$p\tau$



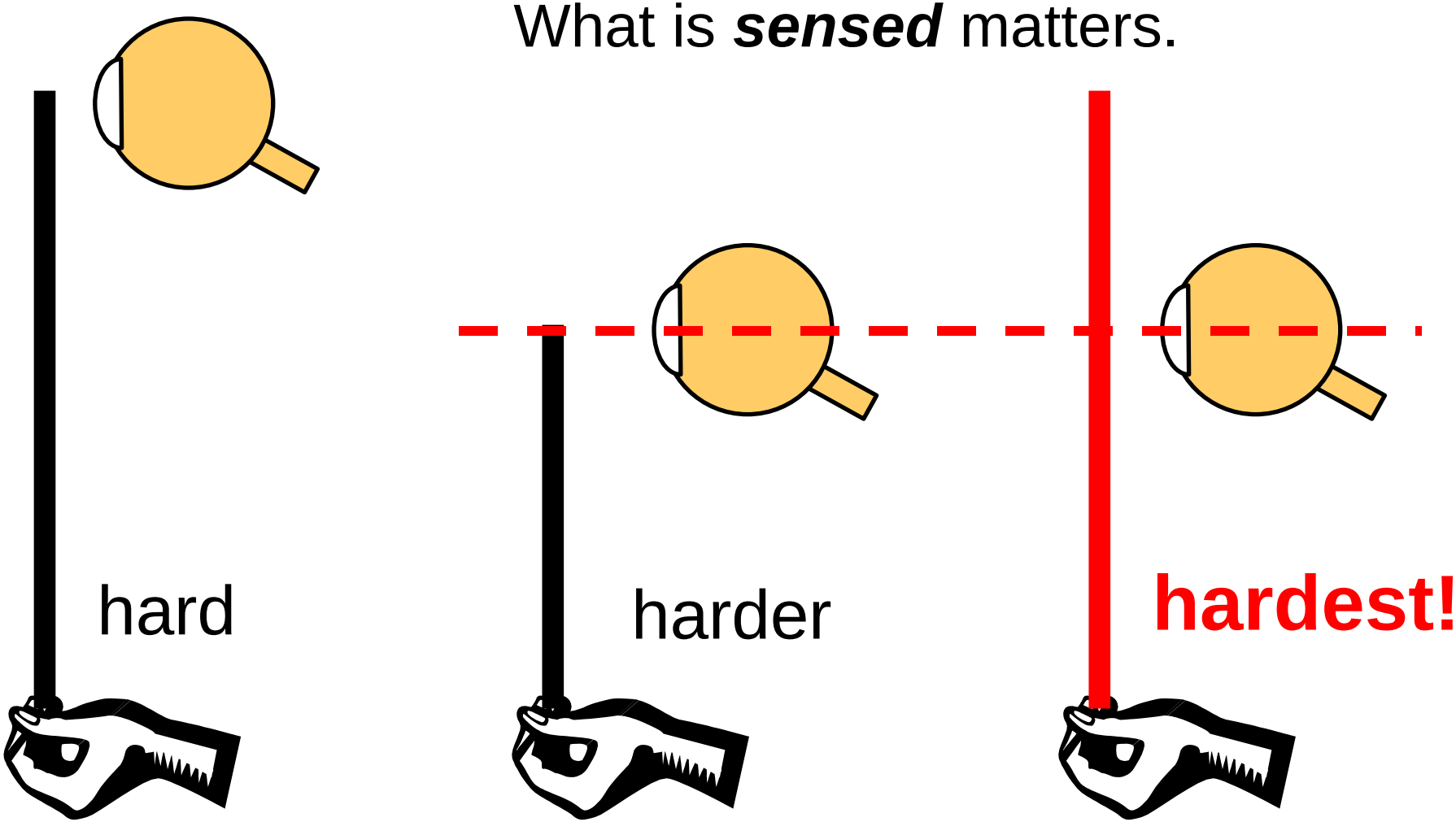
Law #4 : $$\frac{1}{\pi}\int_0^\infty \ln\left|T(j\omega)\right|\left(\frac{2p}{p^2+\omega^2}\right)d\omega \geq p\tau$$

Fragility

$p\tau$



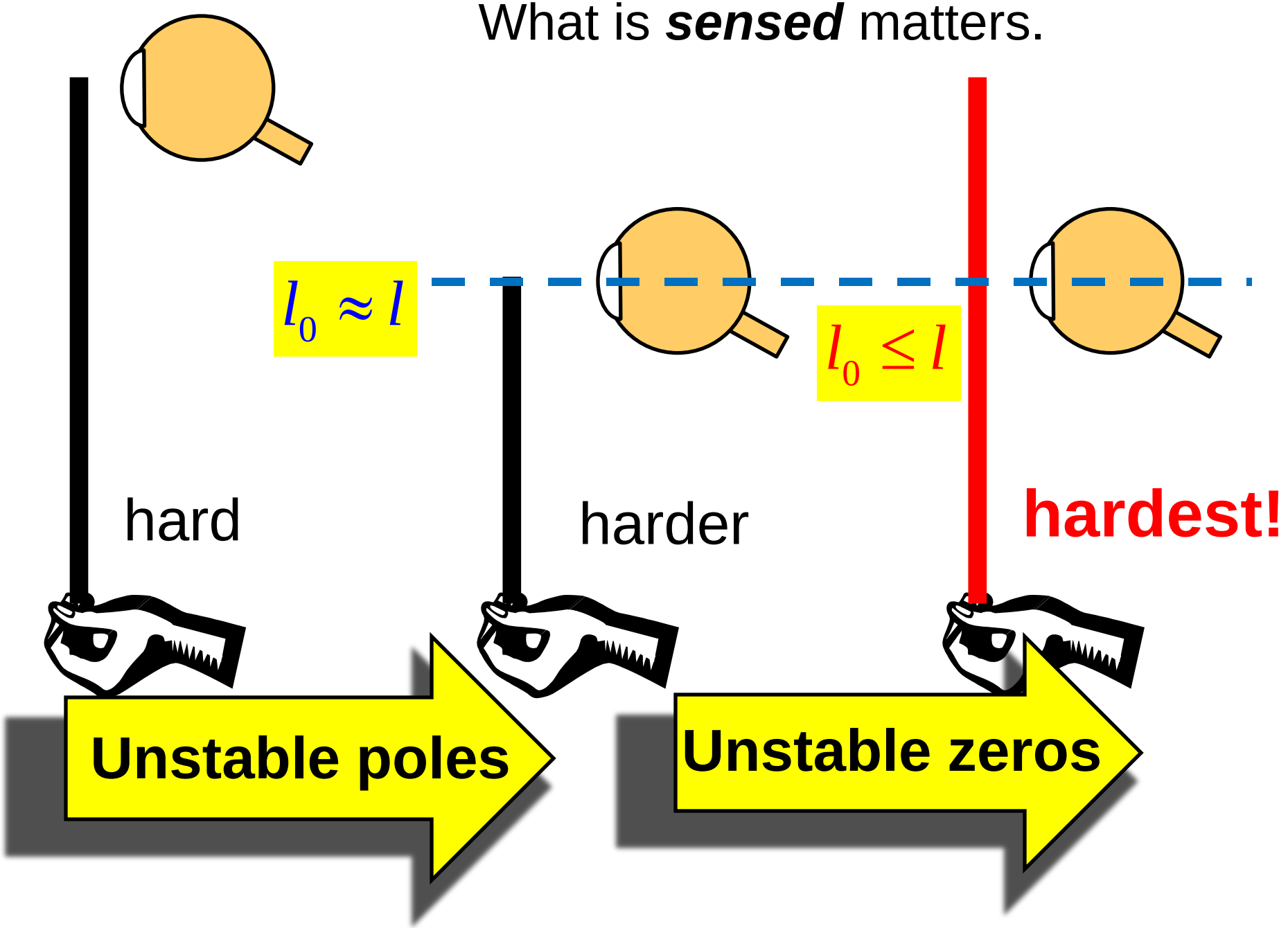
What is ***sensed*** matters.



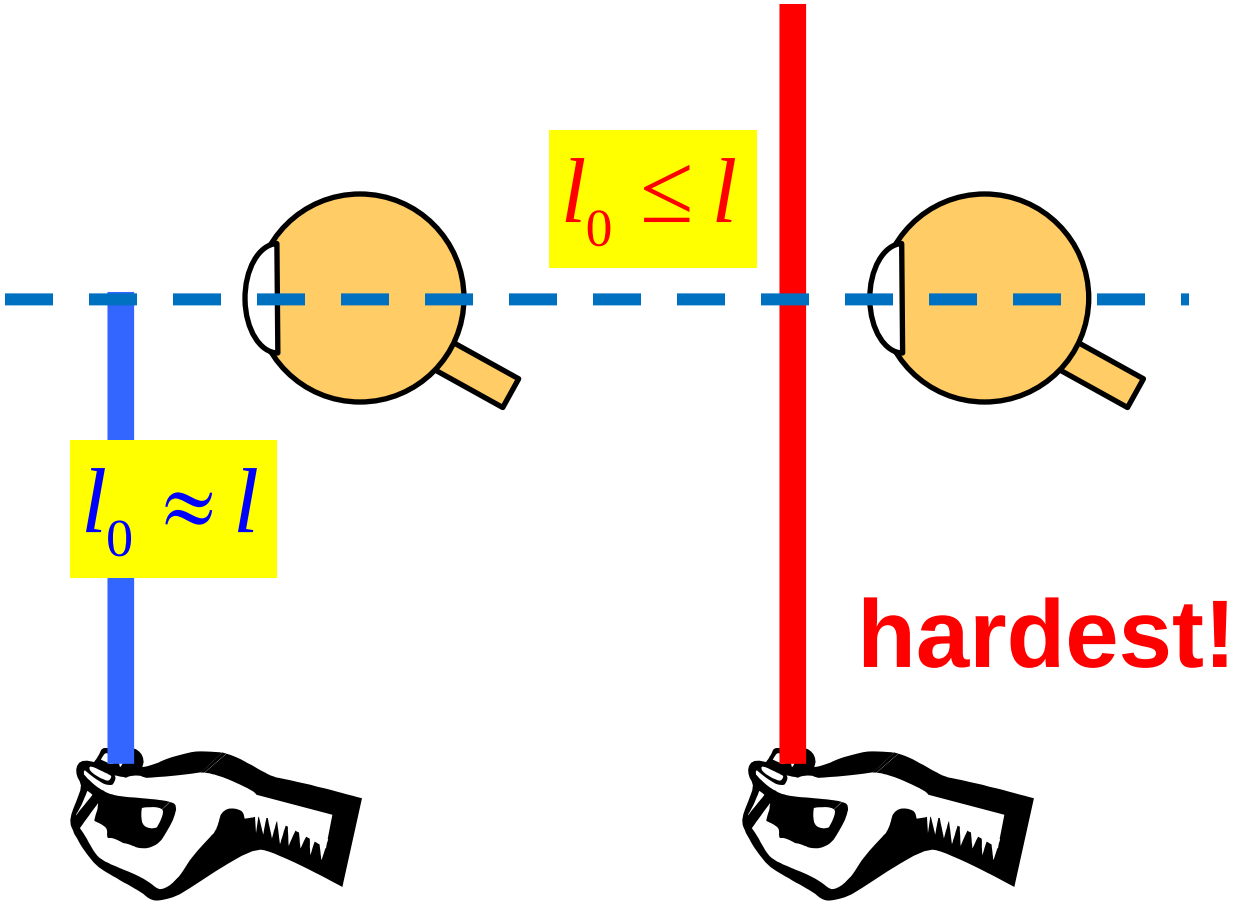
Why?

Easy to ***prove*** using simple models.

What is ***sensed*** matters.



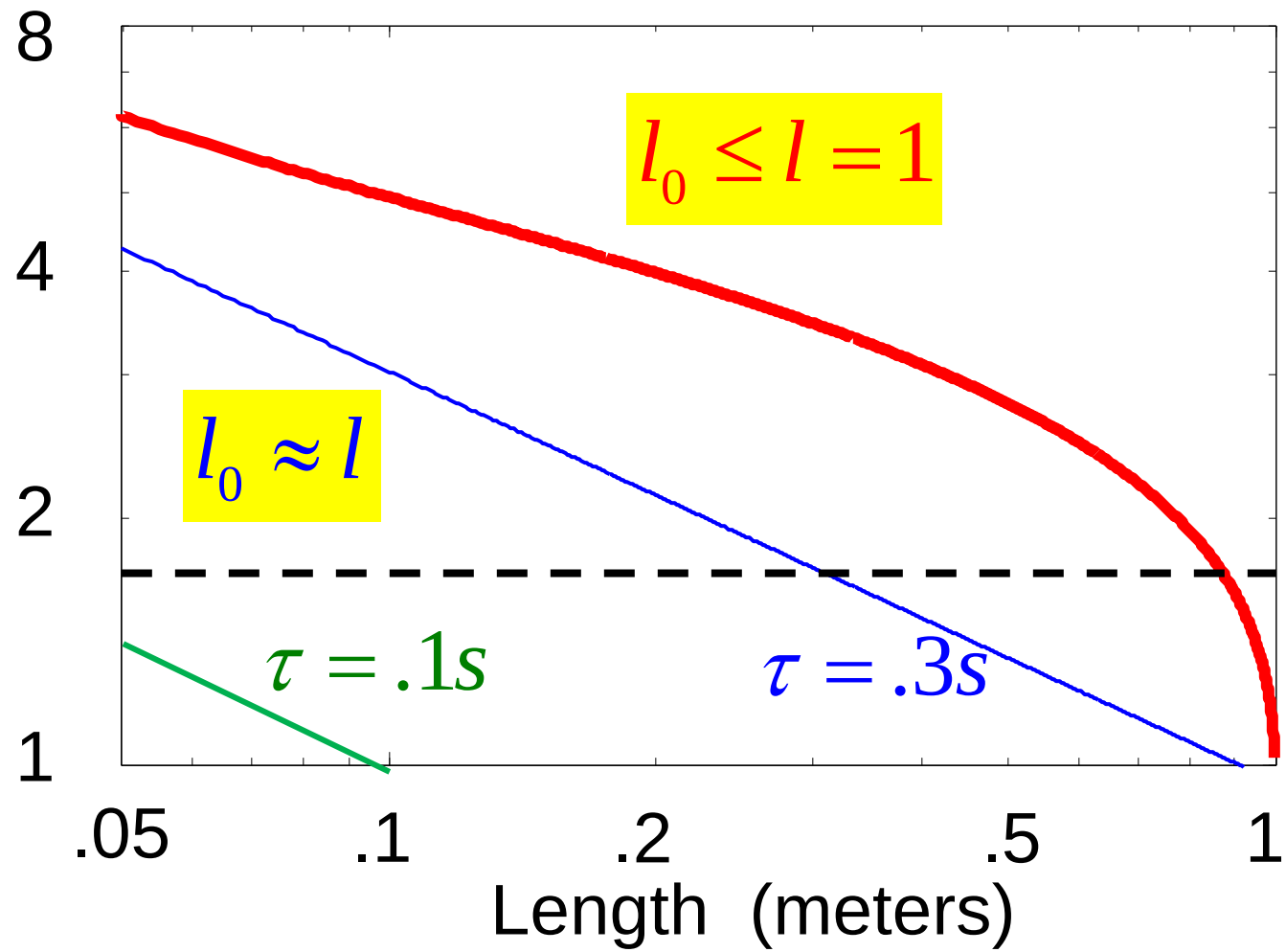
$$\frac{1}{\pi}\int_0^\infty \ln\left|T(j\omega)\right|\left(\frac{2p}{p^2+\omega^2}\right)d\omega \geq p\tau + \ln\left|\frac{z+p}{z-p}\right|$$

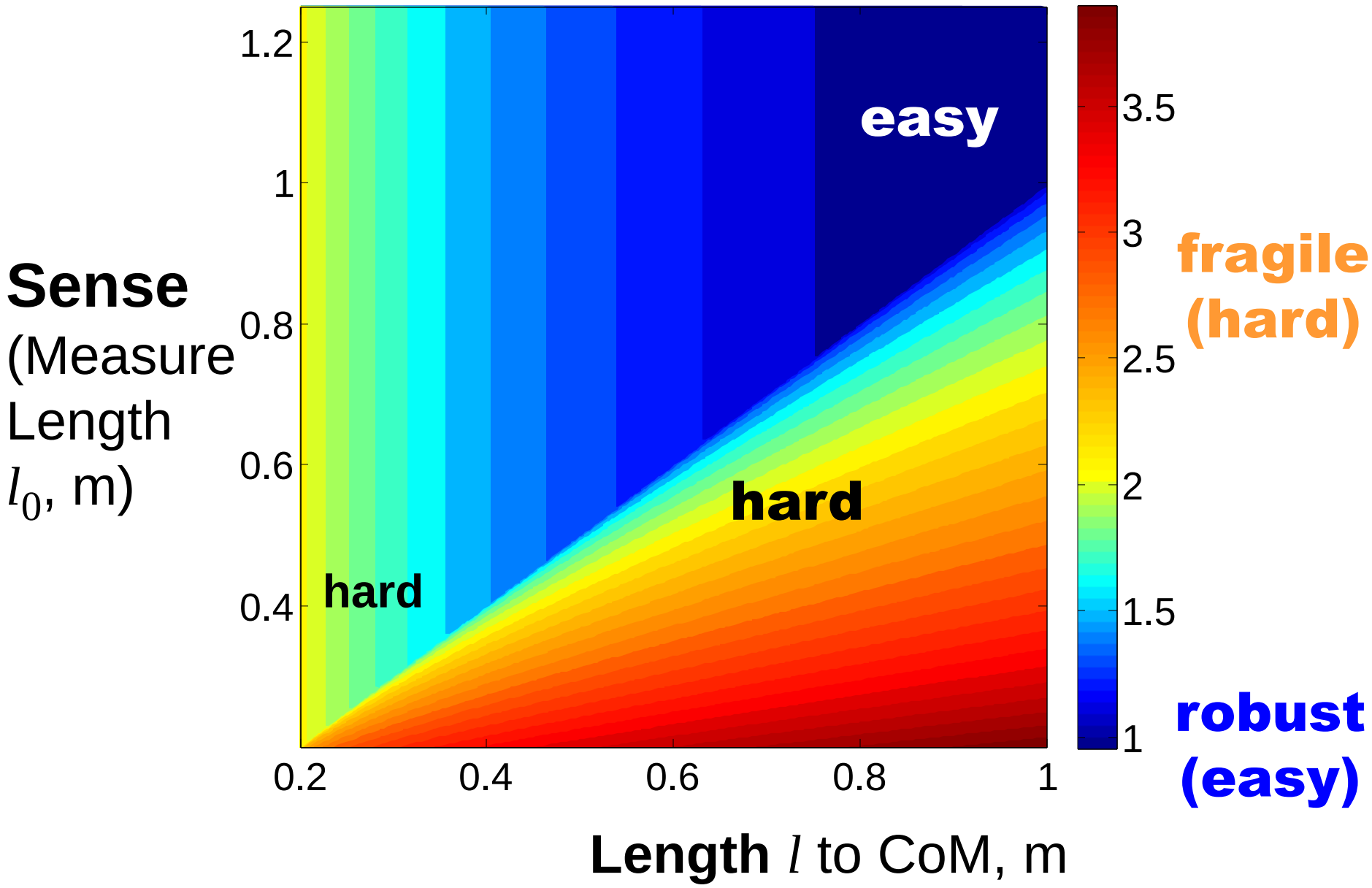


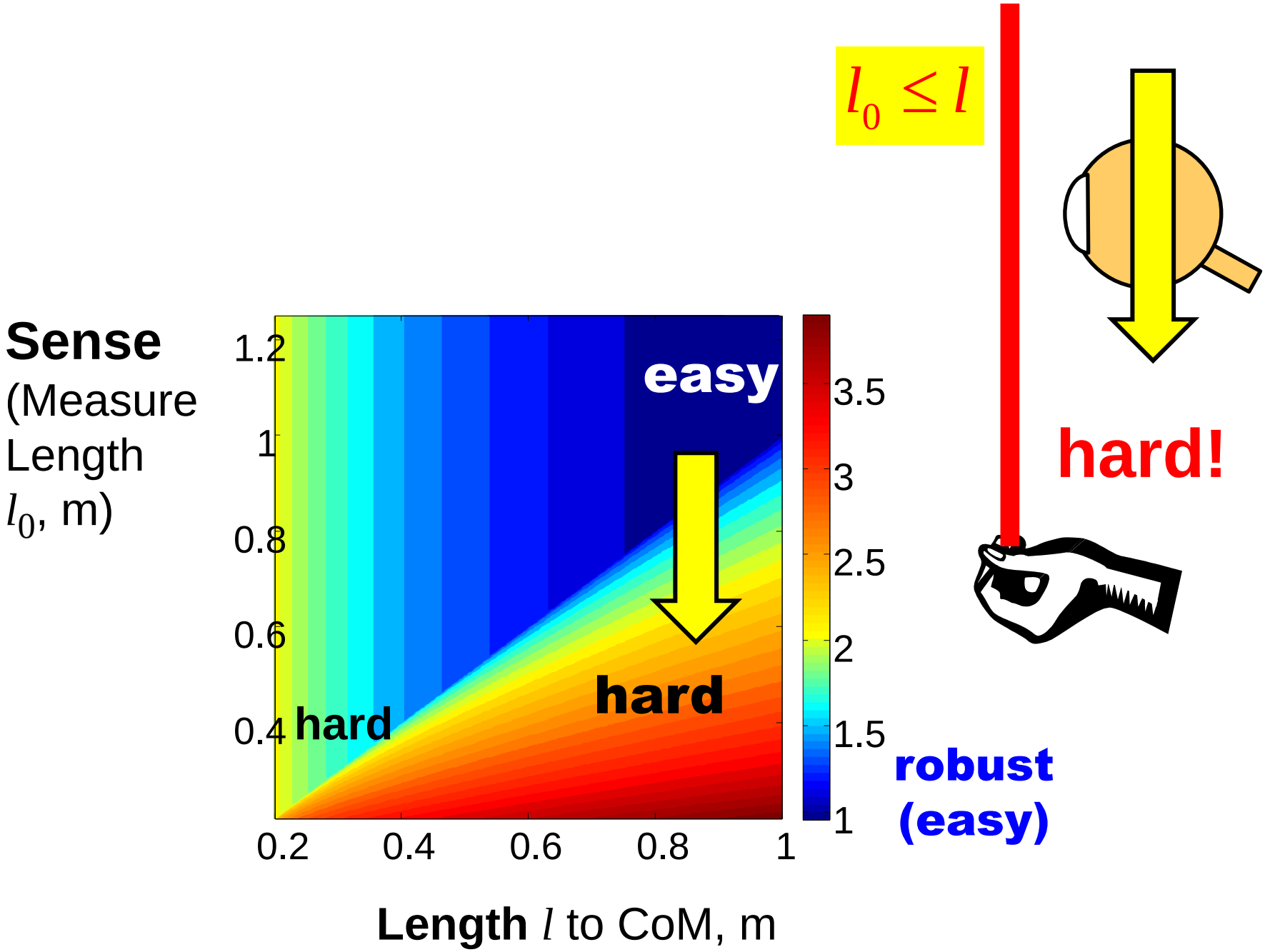
$$\frac{1}{\pi}\int_0^\infty \ln\left|T(j\omega)\right|\left(\frac{2p}{p^2+\omega^2}\right)d\omega \geq p\tau + \ln\left|\frac{z+p}{z-p}\right|$$

Fragility

$$p\tau + \ln\left|\frac{z+p}{z-p}\right|$$

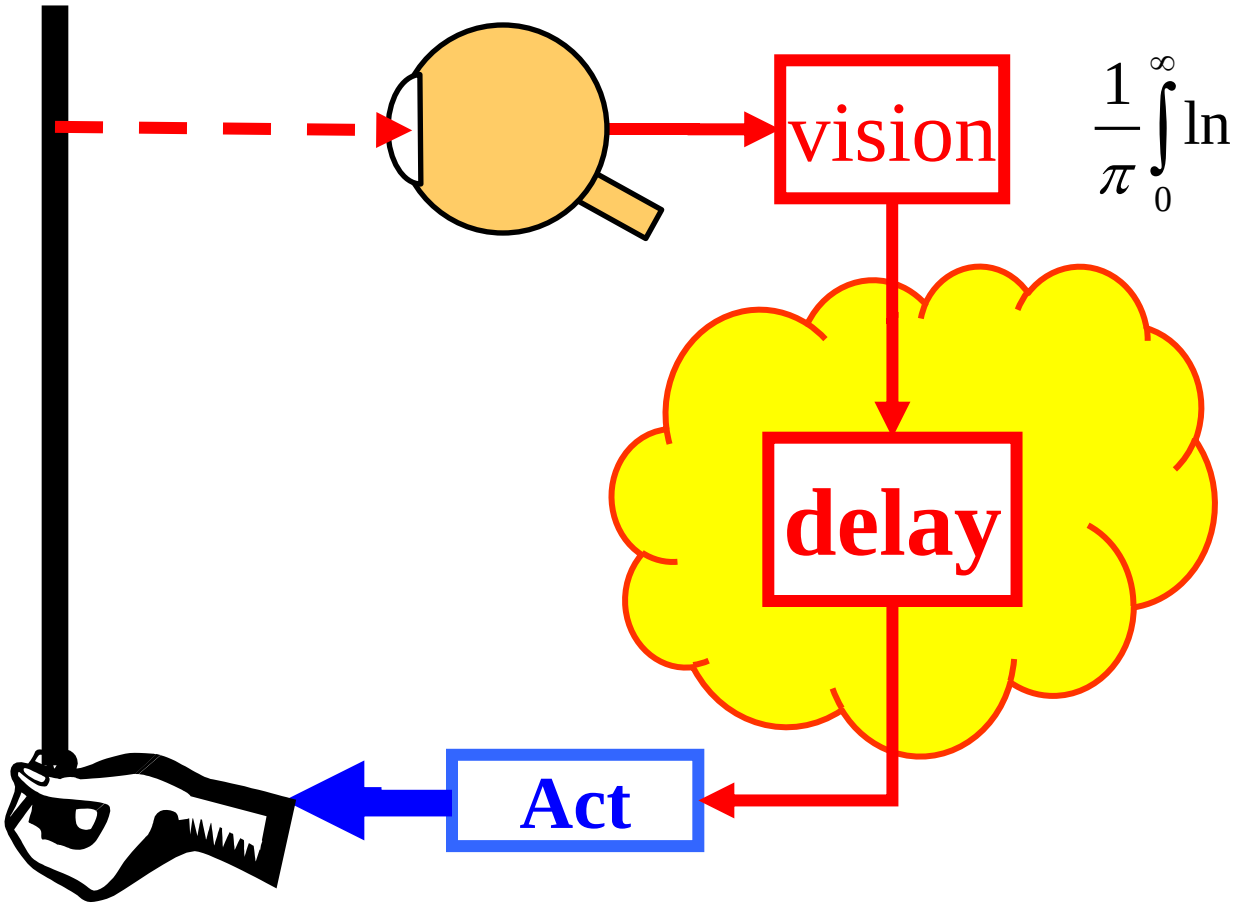



Sense
(Measure
Length
l_0, m)
1.2
1
0.8
0.6
0.4
0.2
0.4
0.6
0.8
1
easy
hard
hard
3.5
3
2.5
2
1.5
1
fragile
(hard)
robust
(easy)
Length l to CoM, m


$l_0 \leq l$
hard!
Sense
(Measure
Length
l_0, m)
easy
hard
hard
robust
(easy)
1.2
1
0.8
0.6
0.4
0.2
0.4
0.6
0.8
1
3.5
3
2.5
2
1.5
1
Length l to CoM, m

Holds for *all* controllers.

Like Turing, a “law” about intrinsic problem difficulty.



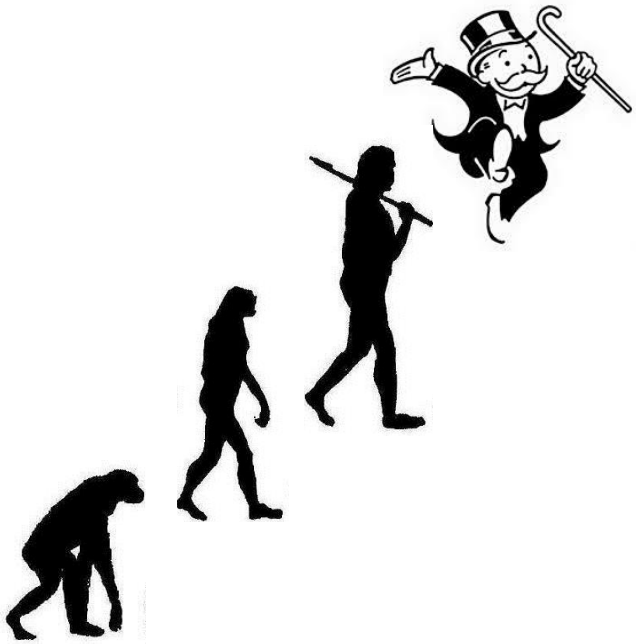
$$\frac{1}{\pi}\int_0^\infty \ln\left|T(j\omega)\right|\left(\frac{p}{p^2+\omega^2}\right)d\omega$$

$$\geq p\tau + \ln\left|\frac{z+p}{z-p}\right|$$

Select instabilities in biology

Working backwards

- Money/finance/lobbyists/etc
- Society/agriculture/weapons/etc
- Bipedalism
- Maternal care
- Warm blood
- Flight
- Mitochondria
- Translation (ribosomes)
- Glycolysis (see 2011 Science paper)



UG biochem, math, control theory

Glycolytic Oscillations and Limits on Robust Efficiency

Fiona A. Chandra,[1]* Gentian Buzi,[2] John C. Doyle[2]

Both engineering and evolution are constrained by trade-offs between efficiency and robustness, but theory that formalizes this fact is limited. For a simple two-state model of glycolysis, we explicitly derive analytic equations for hard trade-offs between robustness and efficiency with oscillations as an inevitable side effect. The model describes how the trade-offs arise from individual parameters, including the interplay of feedback control with autocatalysis of network products necessary to power and catalyze intermediate reactions. We then use control theory to prove that the essential features of these hard trade-off "laws" are universal and fundamental, in that they depend minimally on the details of this system and generalize to the robust efficiency of any autocatalytic network. The theory also suggests worst-case conditions that are consistent with initial experiments.

molecules are consumed upstream and four are produced downstream, which normalizes to $q = 1$ (each y molecule produces two downstream) with kinetic exponent $a = 1$. To highlight essential trade-offs with the simplest possible analysis, we normalize the concentration such that the unperturbed ($\delta = 0$) steady states are $\bar{y} = 1$ and $\bar{x} = 1/k$ [the system can have one additional steady state, which is unstable when $(1, 1/k)$ is stable]. [See the supporting online material (SOM) part I]. The basal rate of the PFK reaction and the consumption rate have been normalized to 1 (the 2 in the numerator and feedback coefficients of the reactions come from these normalizations). Our results hold for more general systems

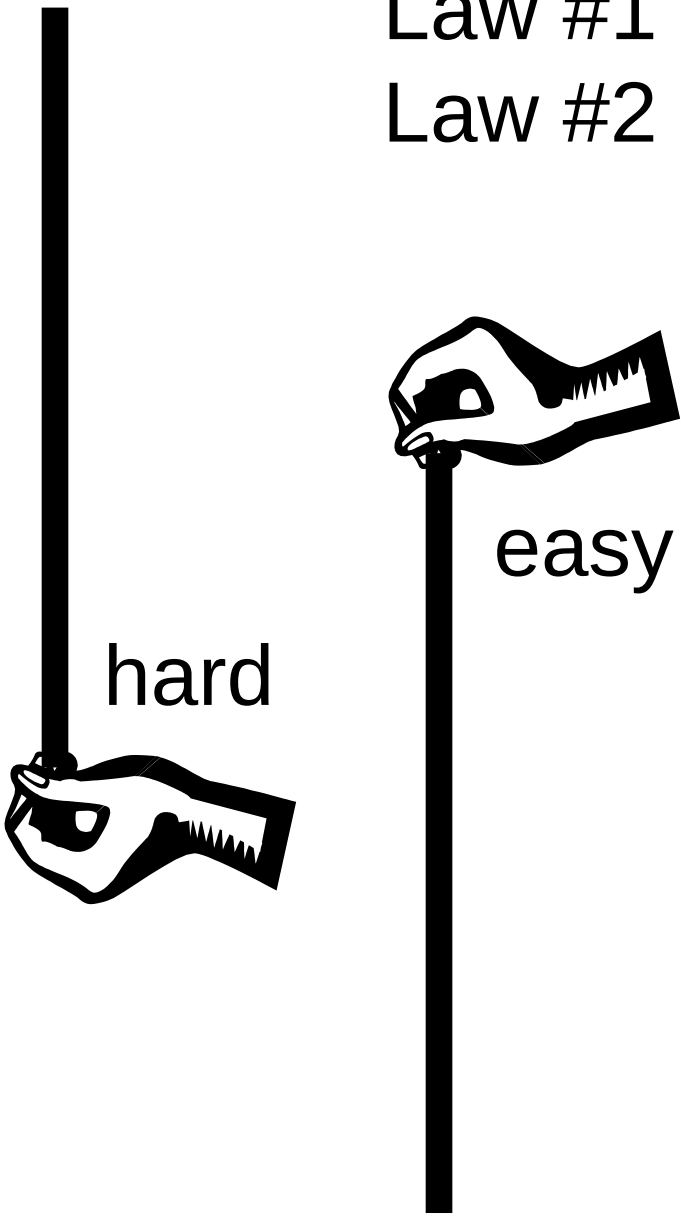
Chandra, Buzi, and Doyle

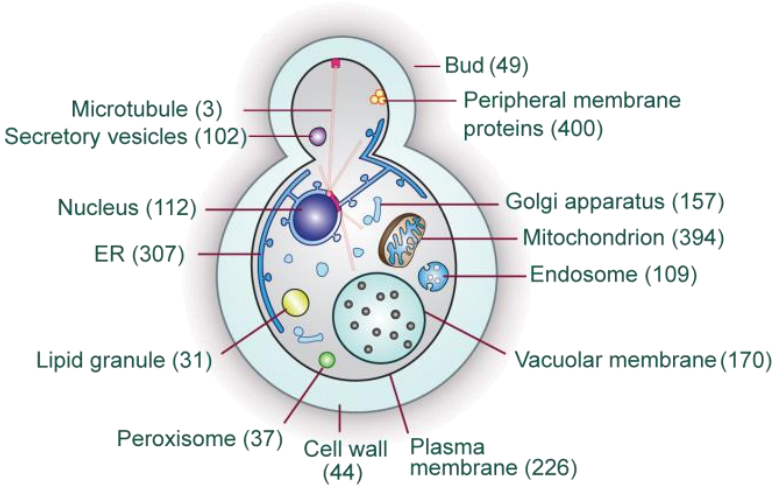
Science AAAS

Most important paper so far.

www.sciencemag.org SCIENCE VOL 333 8 JULY 2011

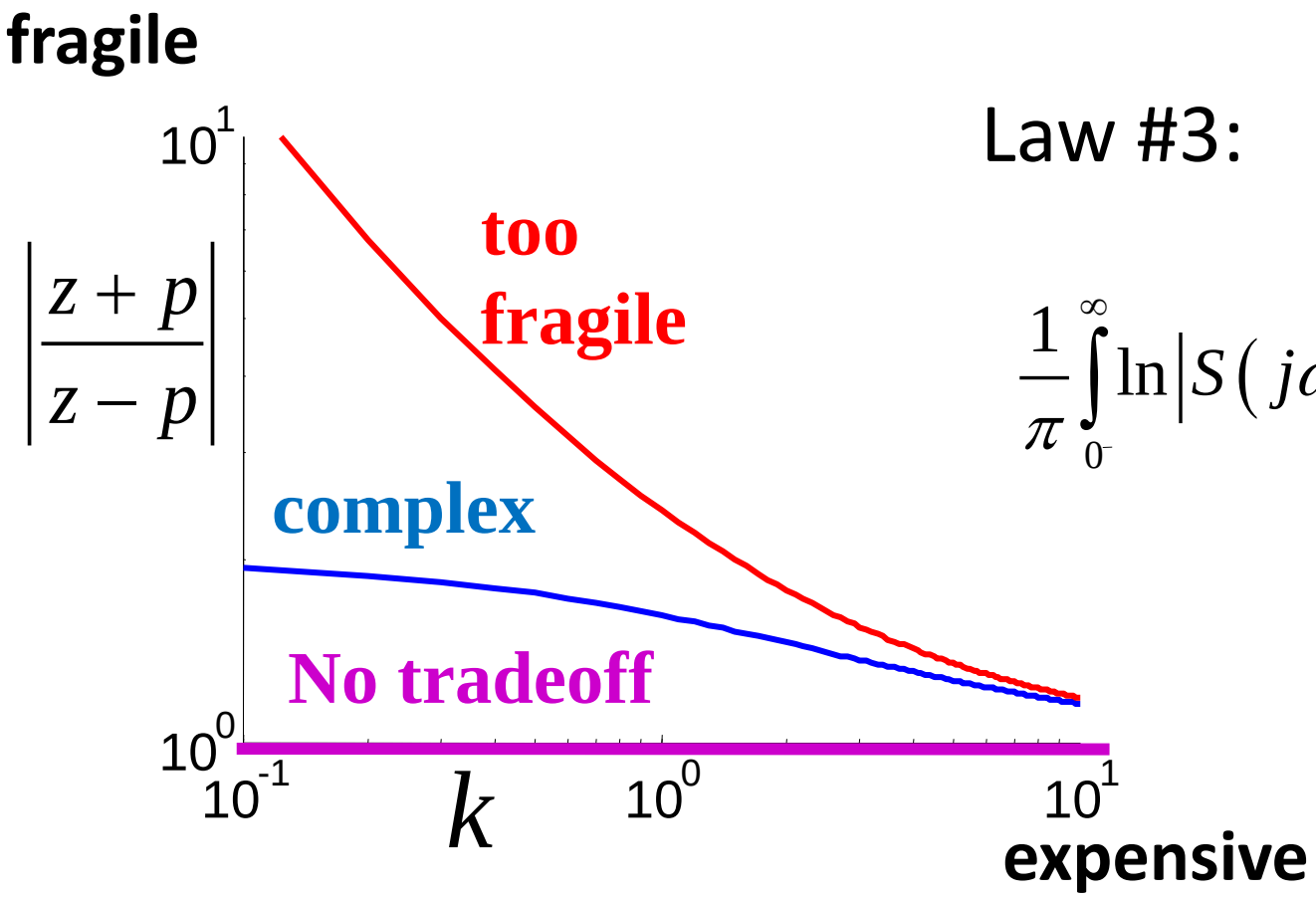
Law #1 : Mechanics ↔ Chemistry
Law #2 : Gravity ↔ Autocatalysis





Law #1 : Chemistry

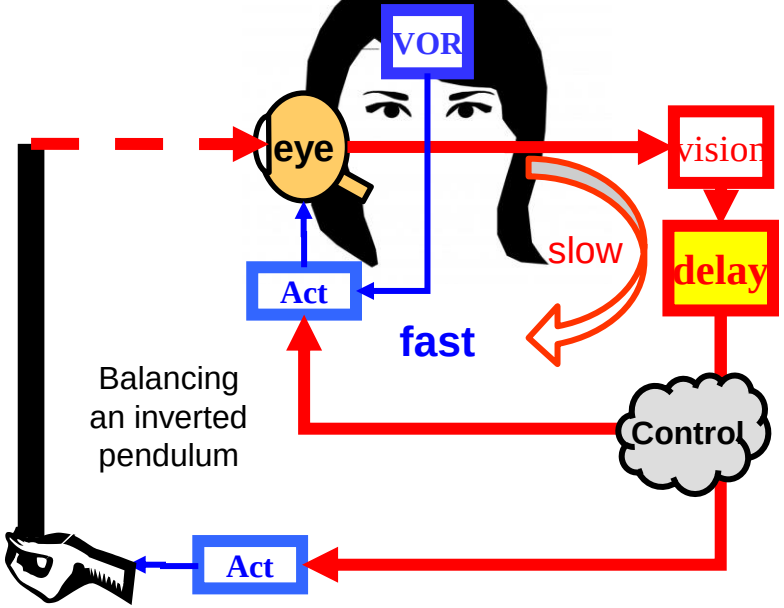
Law #2 : Autocatalysis

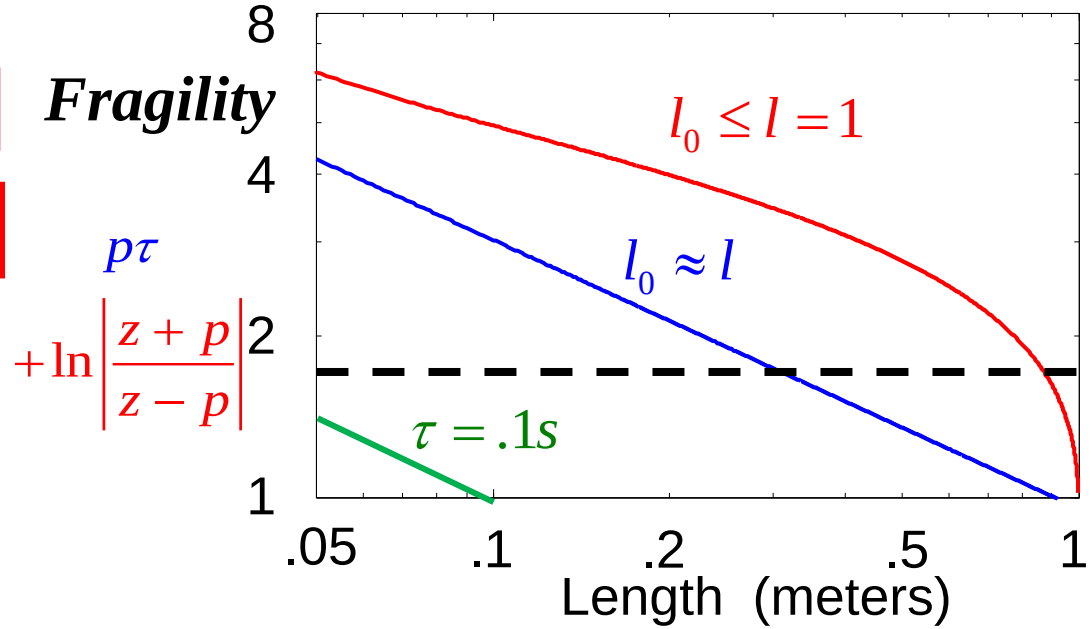


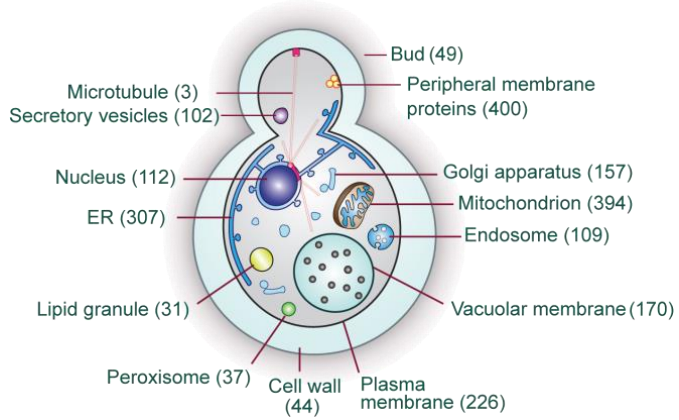
Law #3:

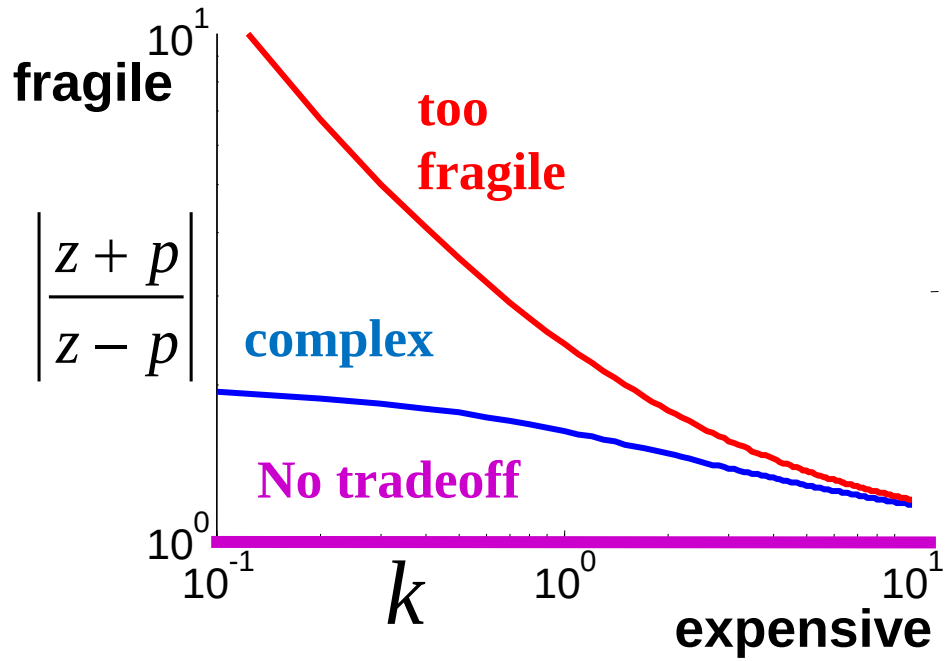
$$\frac{1}{\pi}\int_{0^-}^{\infty} \ln\left|S(j\omega)\right| \left(\frac{z}{z^2+\omega^2}\right) d\omega$$

$$\geq \ln\left|\frac{z+p}{z-p}\right|$$

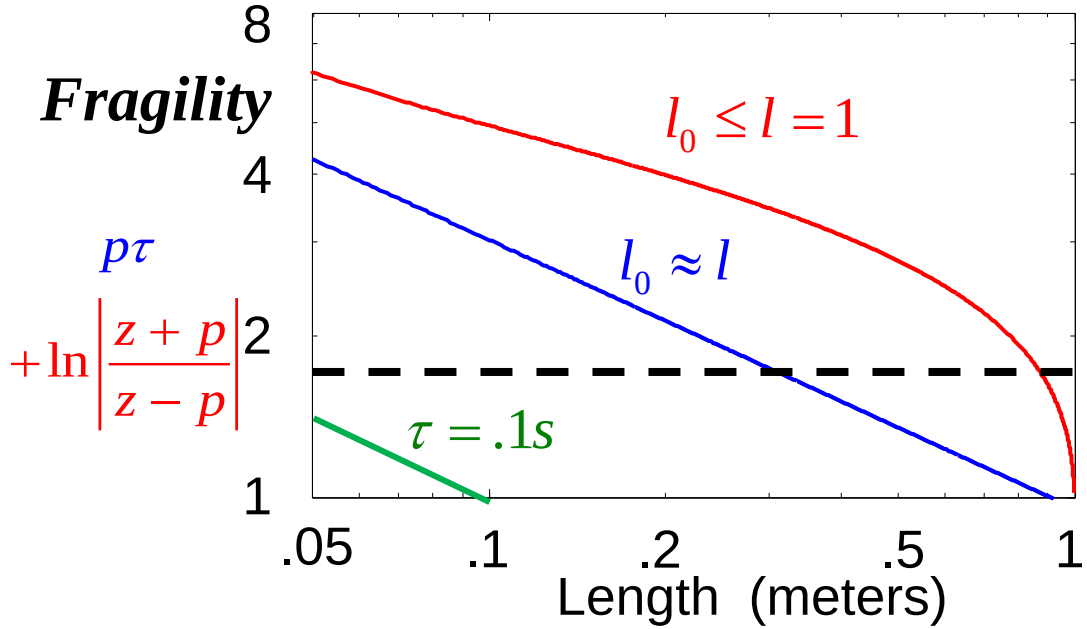

VOR
eye
vision
slow
delay
Act
fast
Balancing
an inverted
pendulum
Control
Act

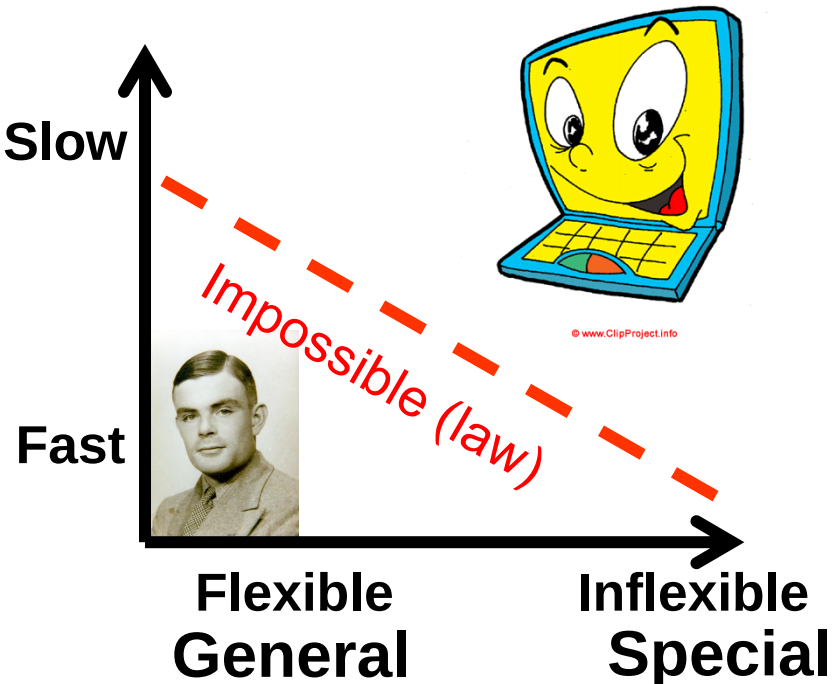

Fragility
$p\tau + \ln\left|\frac{z+p}{z-p}\right|$
$l_0 \le l = 1$
$l_0 \approx l$
$\tau = .1s$
8
4
2
1
.05
.1
.2
.5
1
Length (meters)

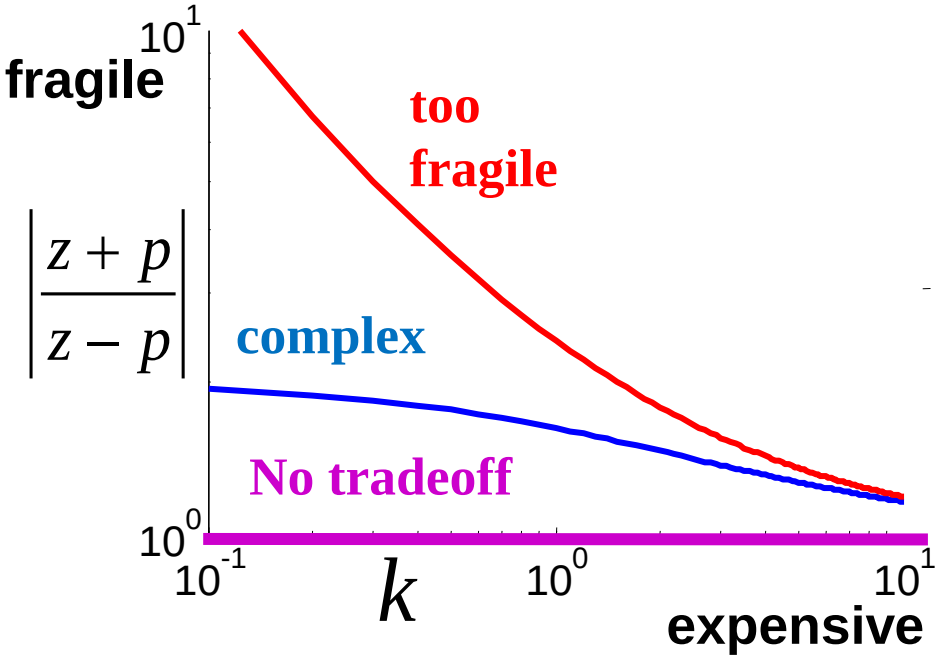

Bud (49)
Microtubule (3)
Peripheral membrane proteins (400)
Secretory vesicles (102)
Nucleus (112)
Golgi apparatus (157)
Mitochondrion (394)
ER (307)
Endosome (109)
Lipid granule (31)
Vacuolar membrane (170)
Peroxisome (37)
Cell wall (44)
Plasma membrane (226)

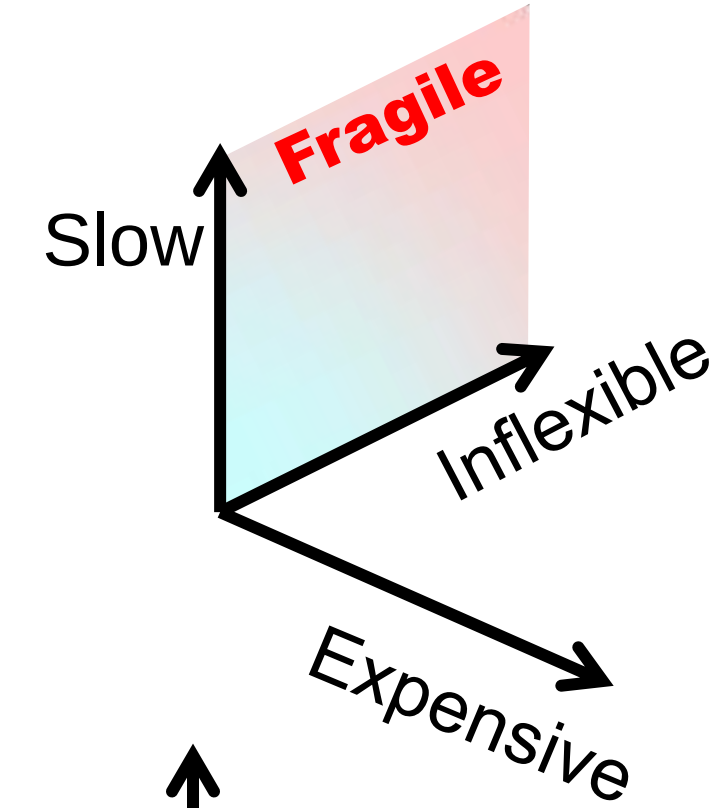

fragile
10^1
too fragile
$\left|\frac{z+p}{z-p}\right|$
complex
No tradeoff
10^0
10^{-1}
k
10^0
10^1
expensive

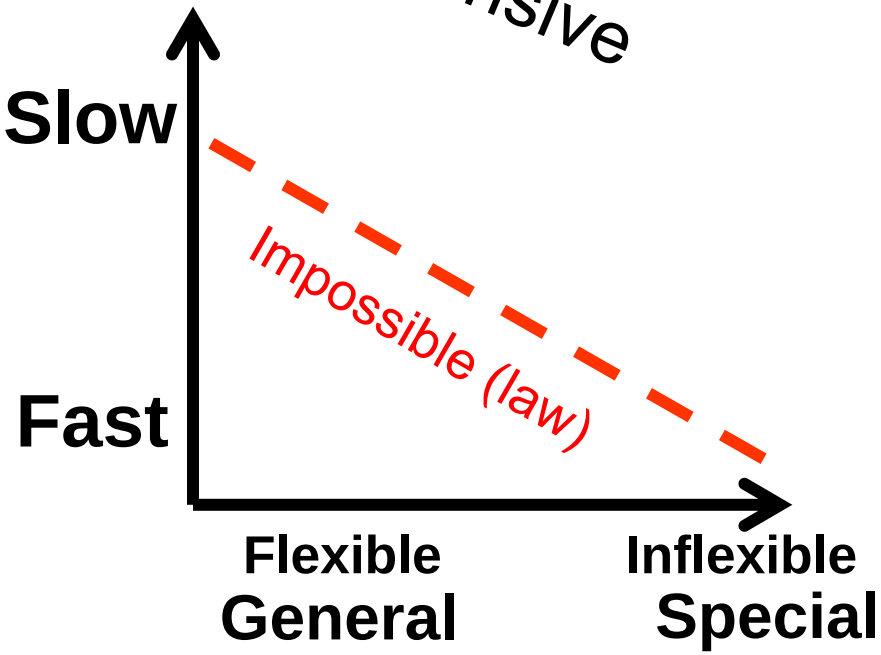
Universal laws

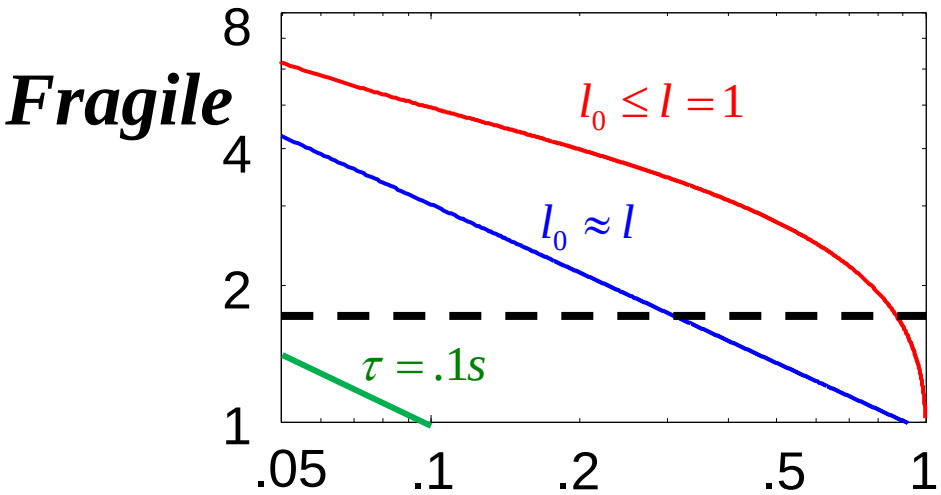


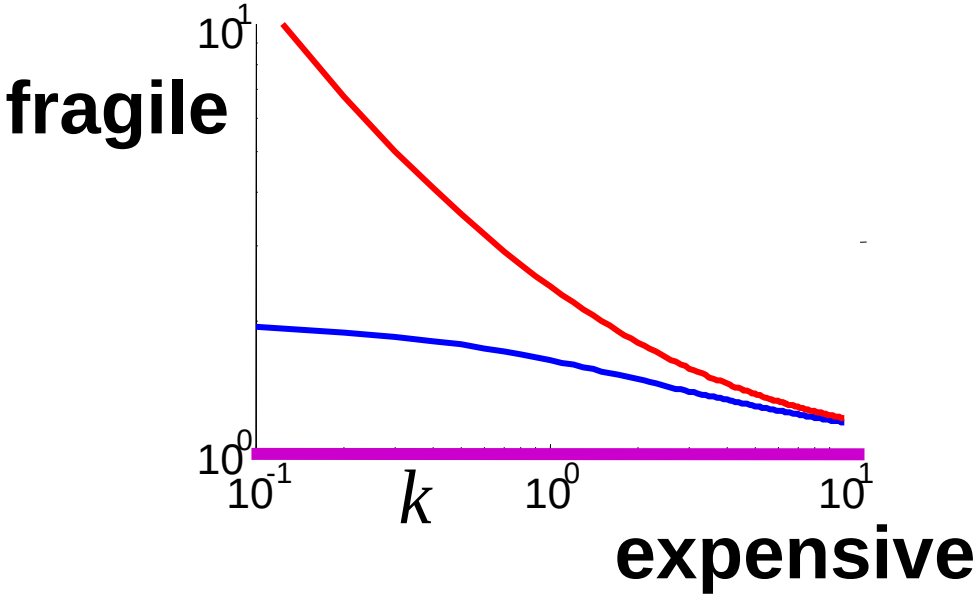




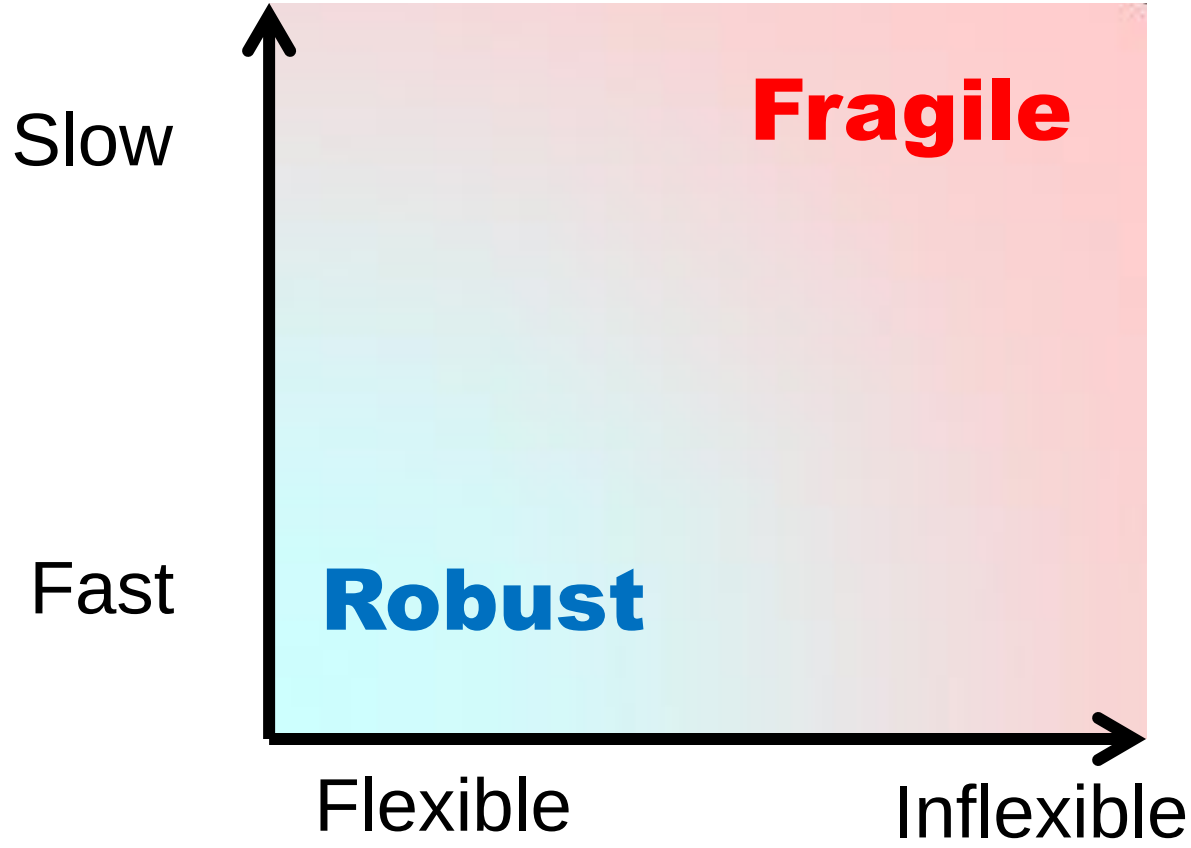

Fragile
Slow
Inflexible
Expensive

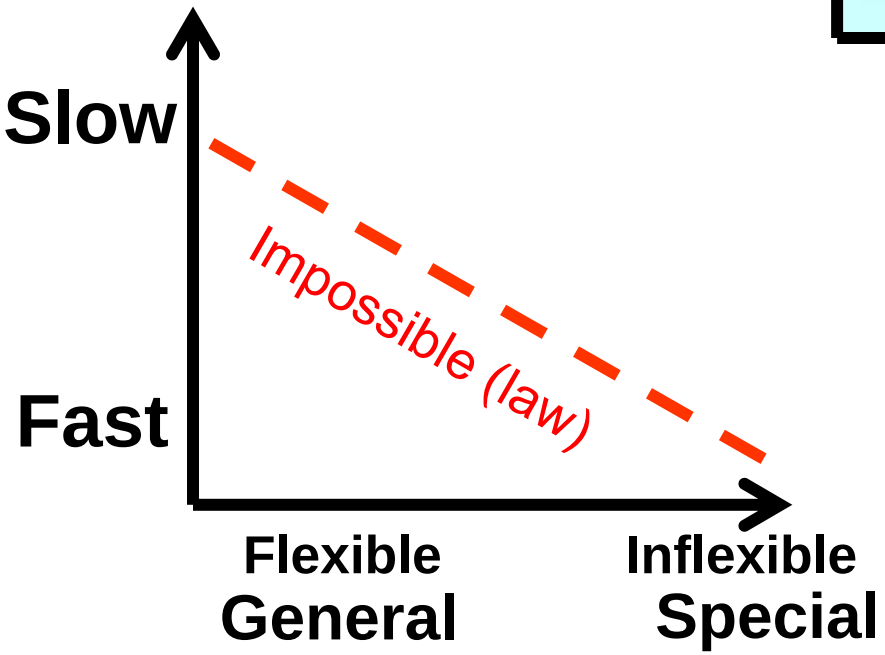

Slow
Fast
Impossible (law)
Flexible
General
Inflexible
Special

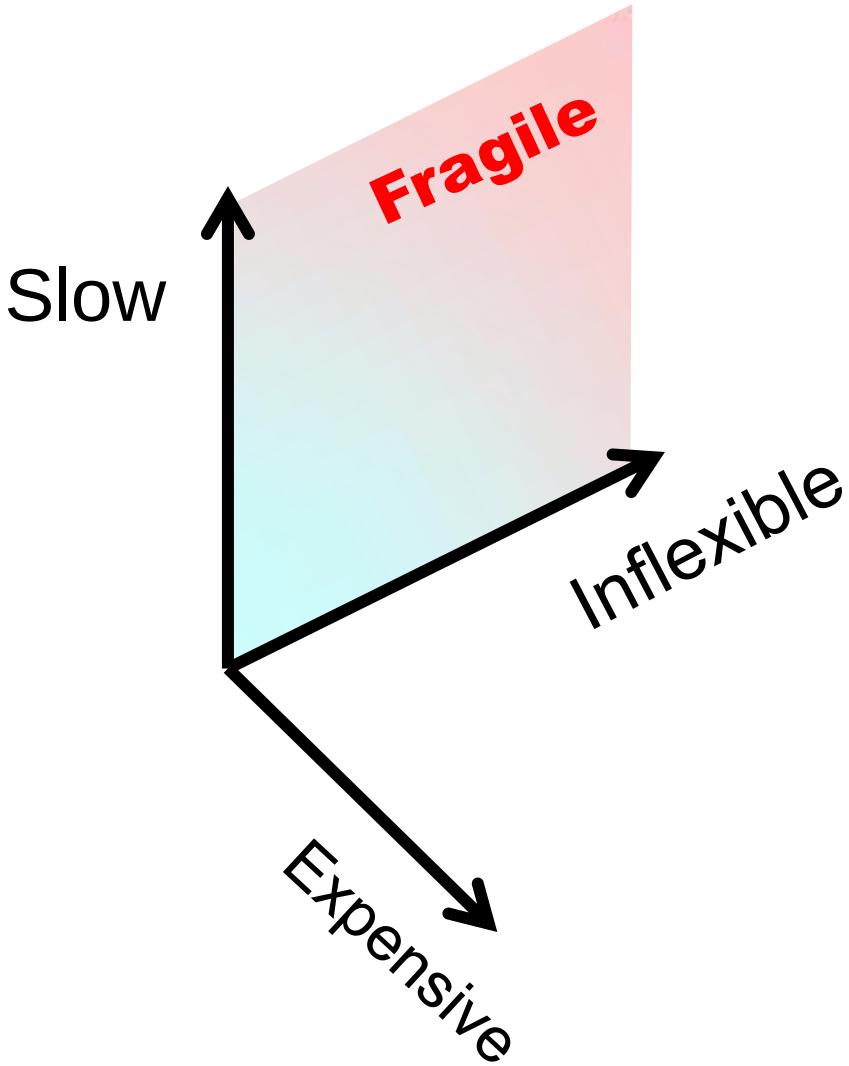

Fragile
8
4
2
1
$l_0 \le l = 1$
$l_0 \approx l$
$\tau = .1s$
.05
.1
.2
.5
1

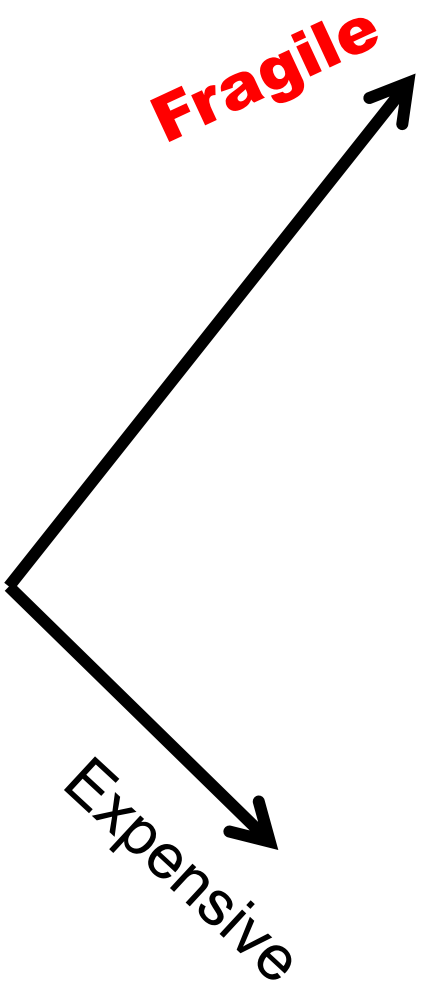

fragile
10^1
10^0
10^{-1}
k
10^0
10^1
expensive

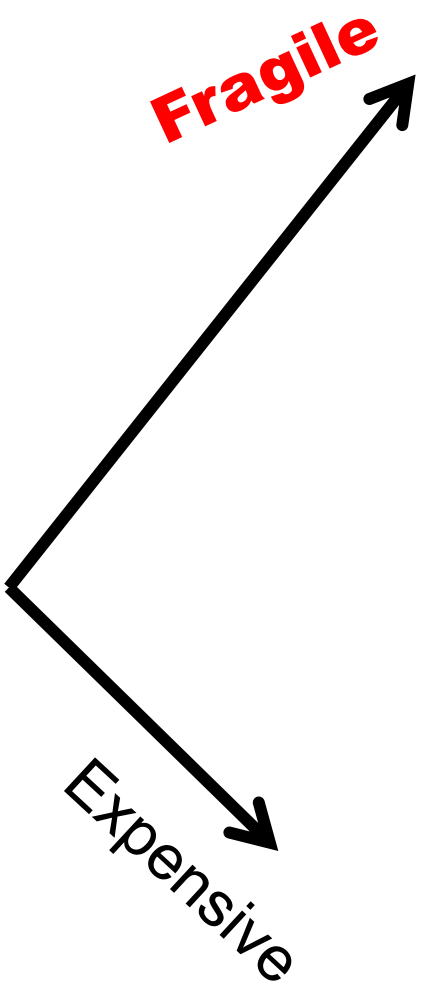
Universal laws

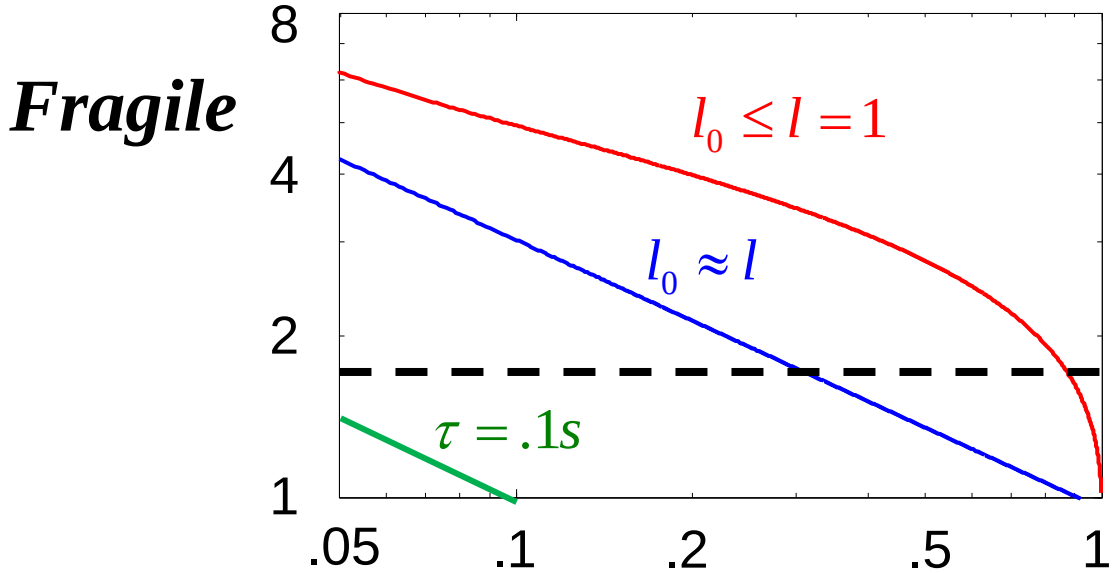


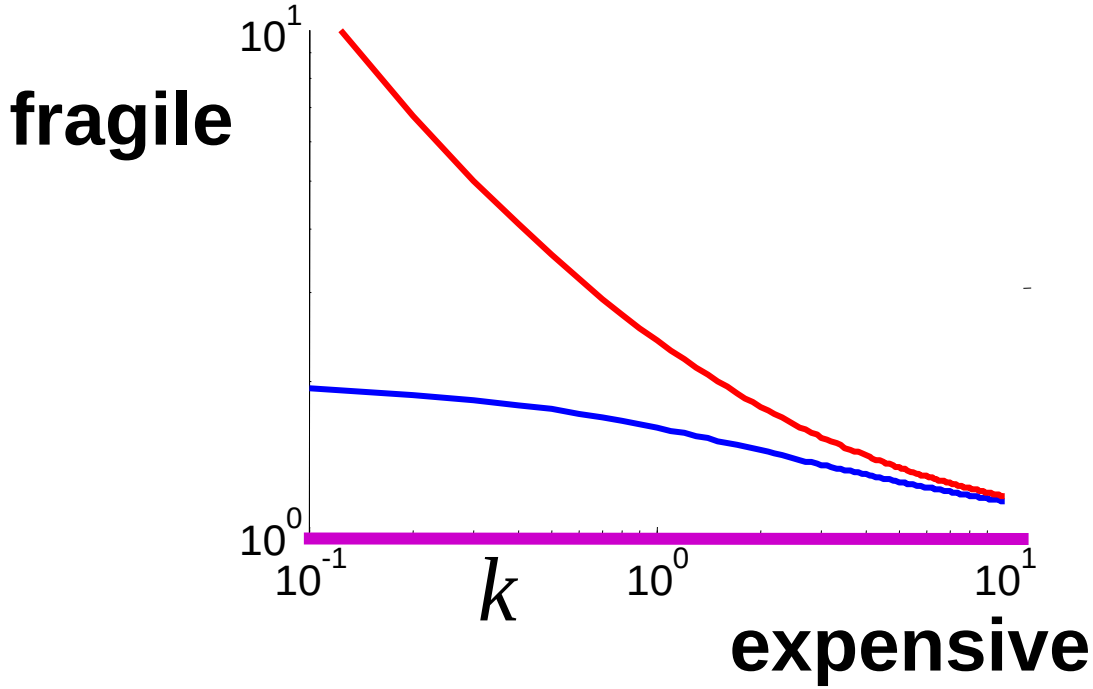


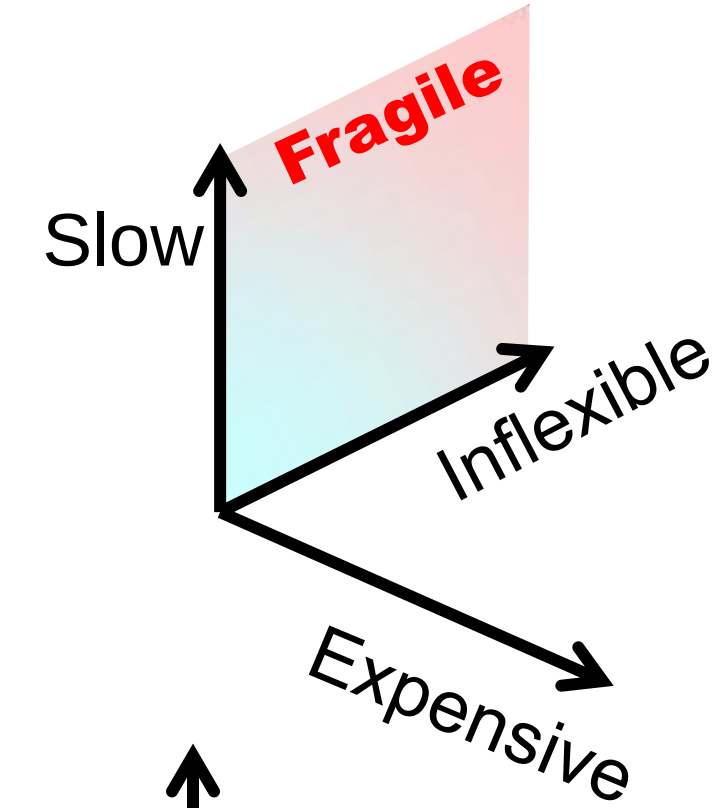

Fragile
Slow
Inflexible
Expensive

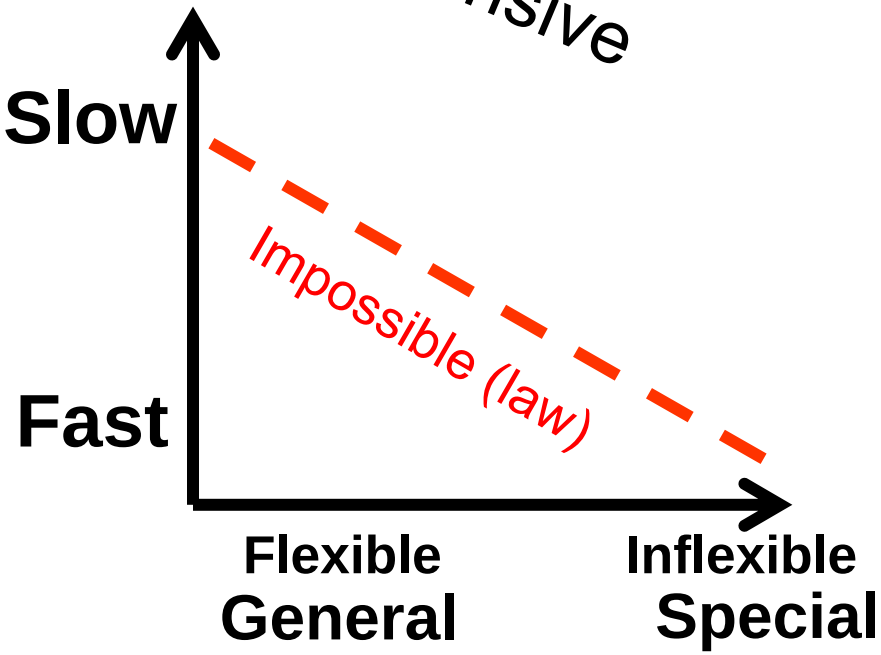

Fragile
Expensive

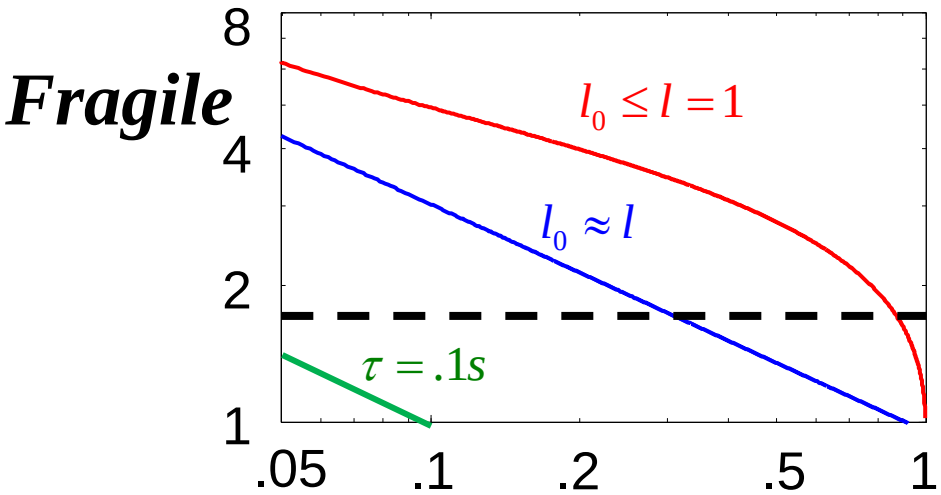

Fragile
Expensive

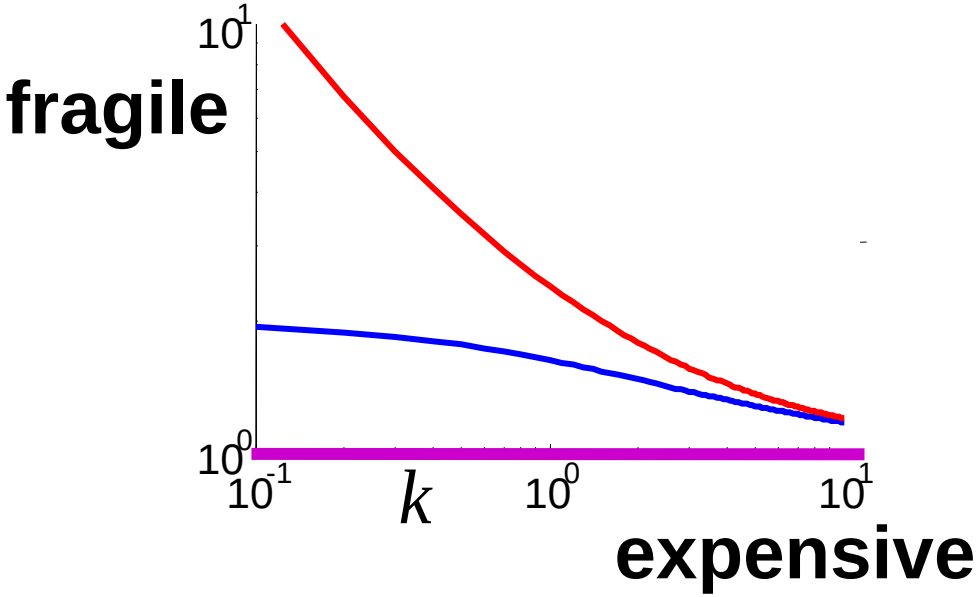

Fragile
8
4
2
1
.05
.1
.2
.5
1
$l_0 \le l = 1$
$l_0 \approx l$
$\tau = .1s$


fragile
10^1
10^0
10^{-1}
k
10^0
10^1
expensive


Fragile
Slow
Inflexible
Expensive

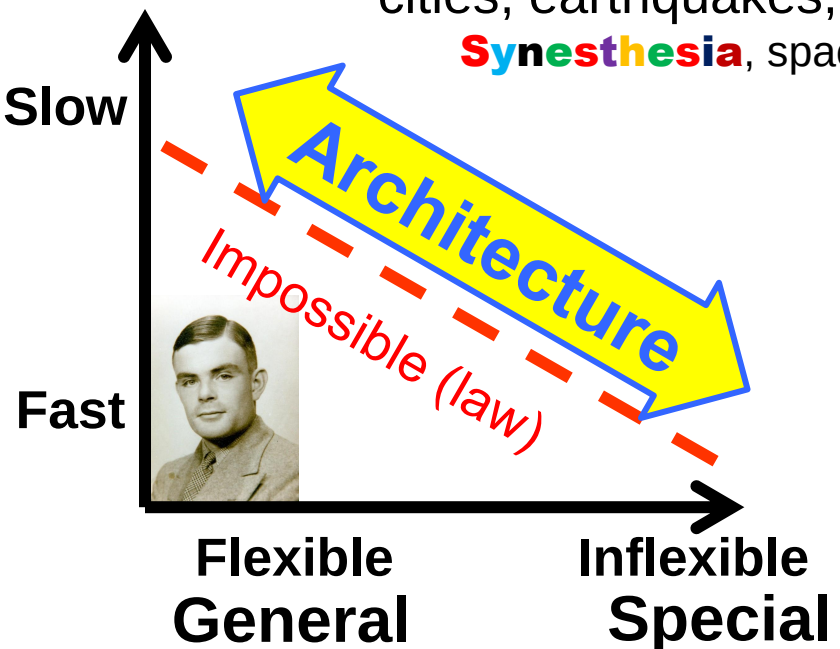

Slow
Fast
Impossible (law)
Flexible
General
Inflexible
Special


Fragile
8
4
2
1
$l_0 \le l = 1$
$l_0 \approx l$
$\tau = .1s$
.05
.1
.2
.5
1

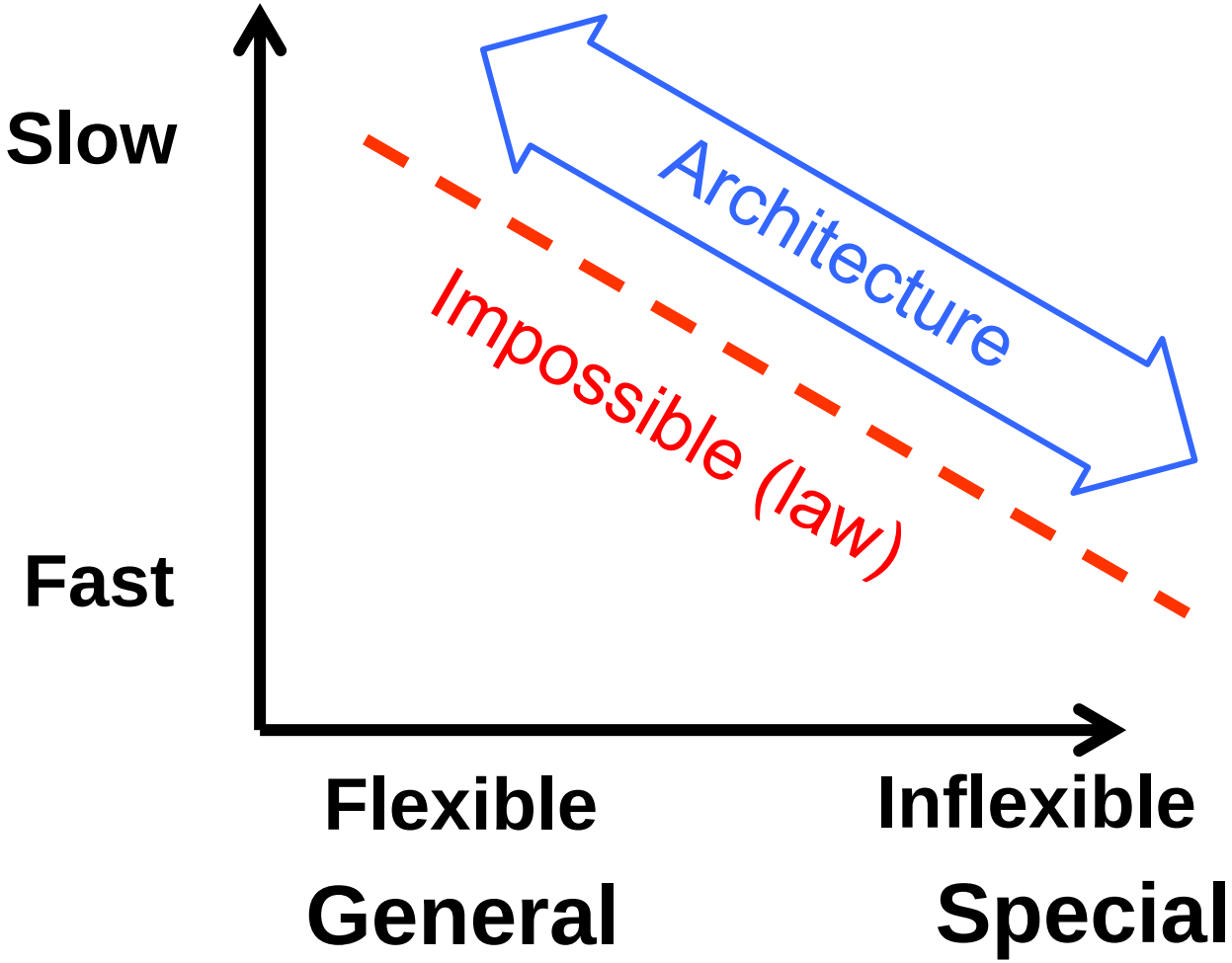

fragile
10^1
10^0
10^{-1}
k
10^0
10^1
expensive

Universal laws and architectures:

nets, grids, brains, bugs,
bodies, doctors, fire, fashion,
art, turbulence, music, buildings,
cities, earthquakes, planes, running, throwing,
Synesthesia, spacecraft, statistical mechanics



Universal Turing Machine

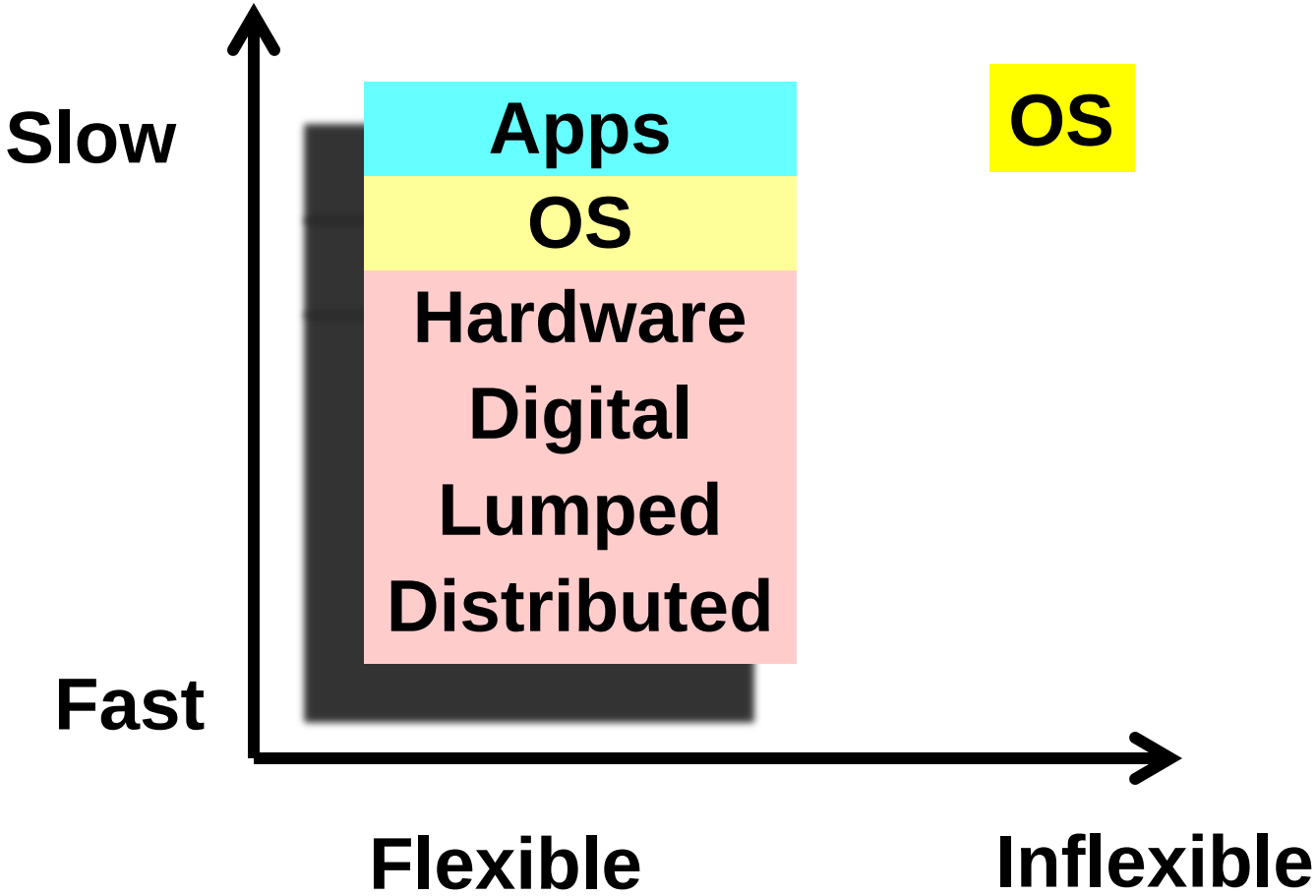


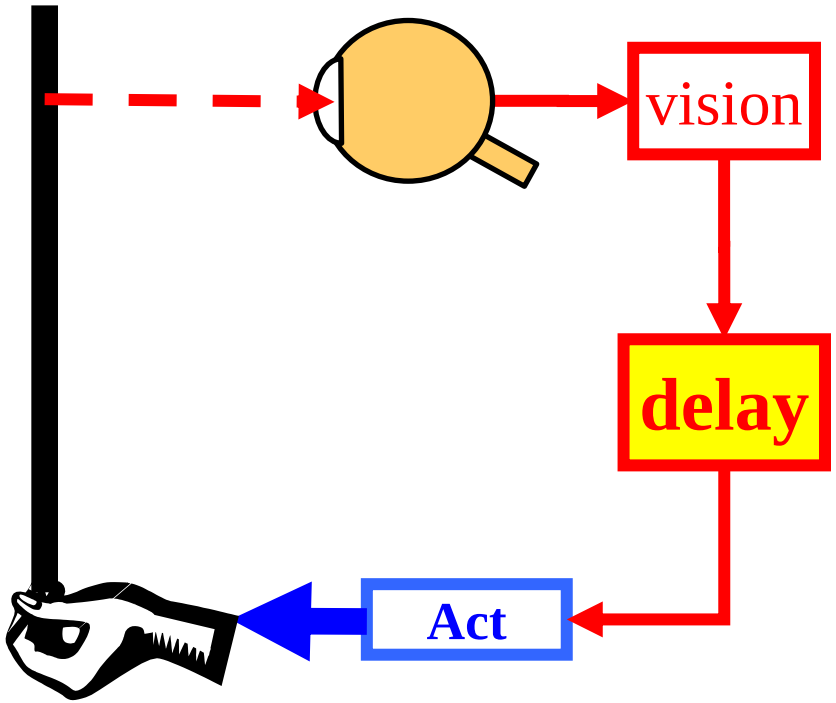
Tech implications/extensions

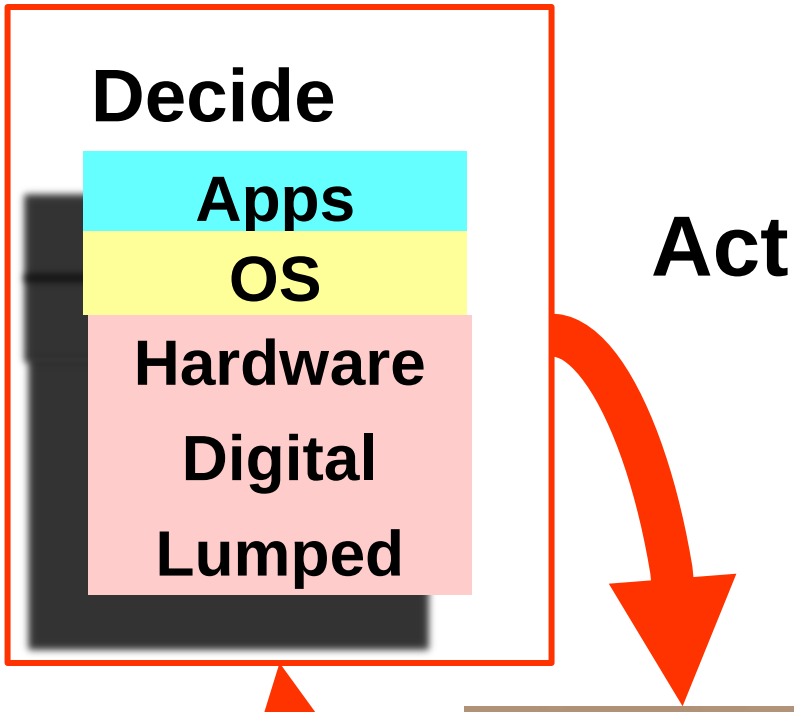
Apps

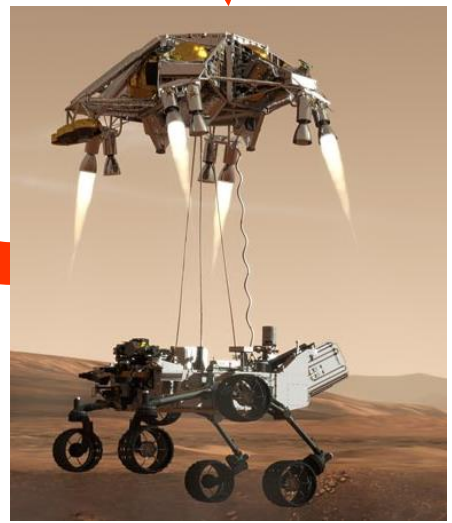
OS

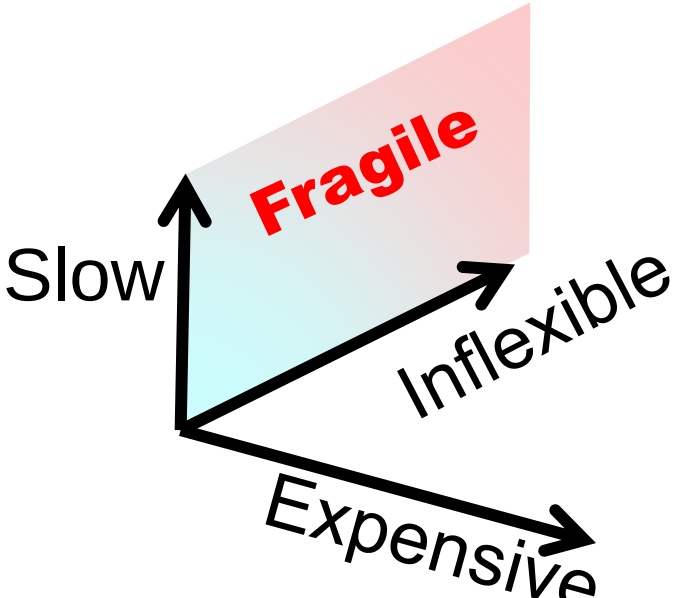
HW




vision
delay
Act

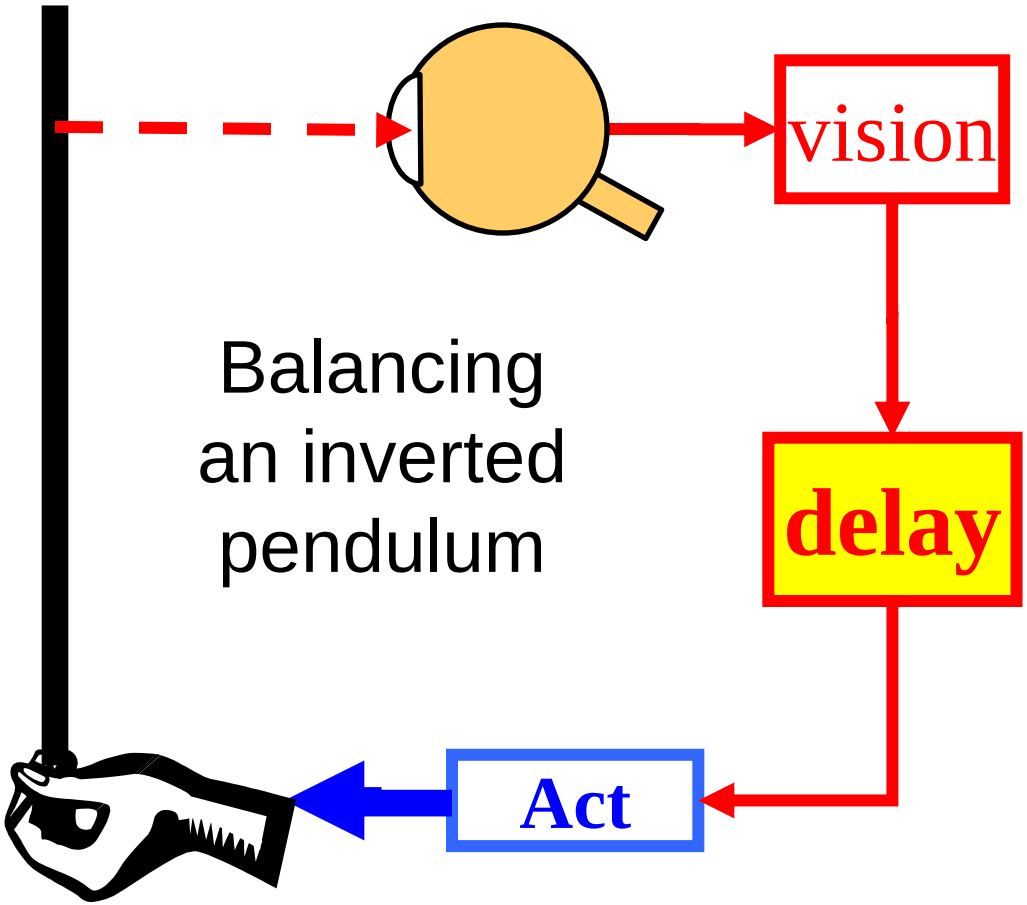

Decide
Apps
OS
Hardware
Digital
Lumped
Act
Sense




Fragile
Slow
Inflexible
Expensive

Completing the story

Mechanics+
Gravity +
Light +



Balancing
an inverted
pendulum

$$\frac{1}{\pi}\int_0^\infty \ln\left|T(j\omega)\right|\left(\frac{p}{p^2+\omega^2}\right)d\omega$$

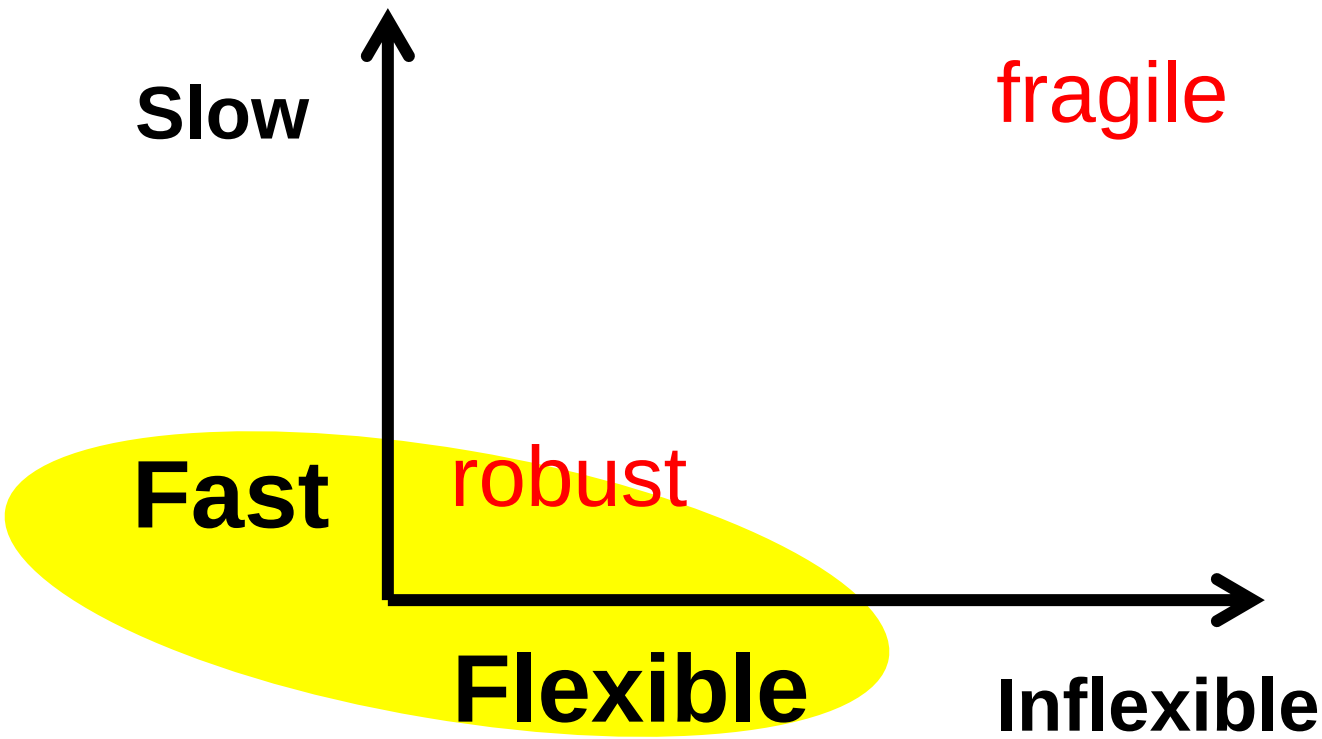
$$\geq p\tau + \ln\left|\frac{z+p}{z-p}\right|$$

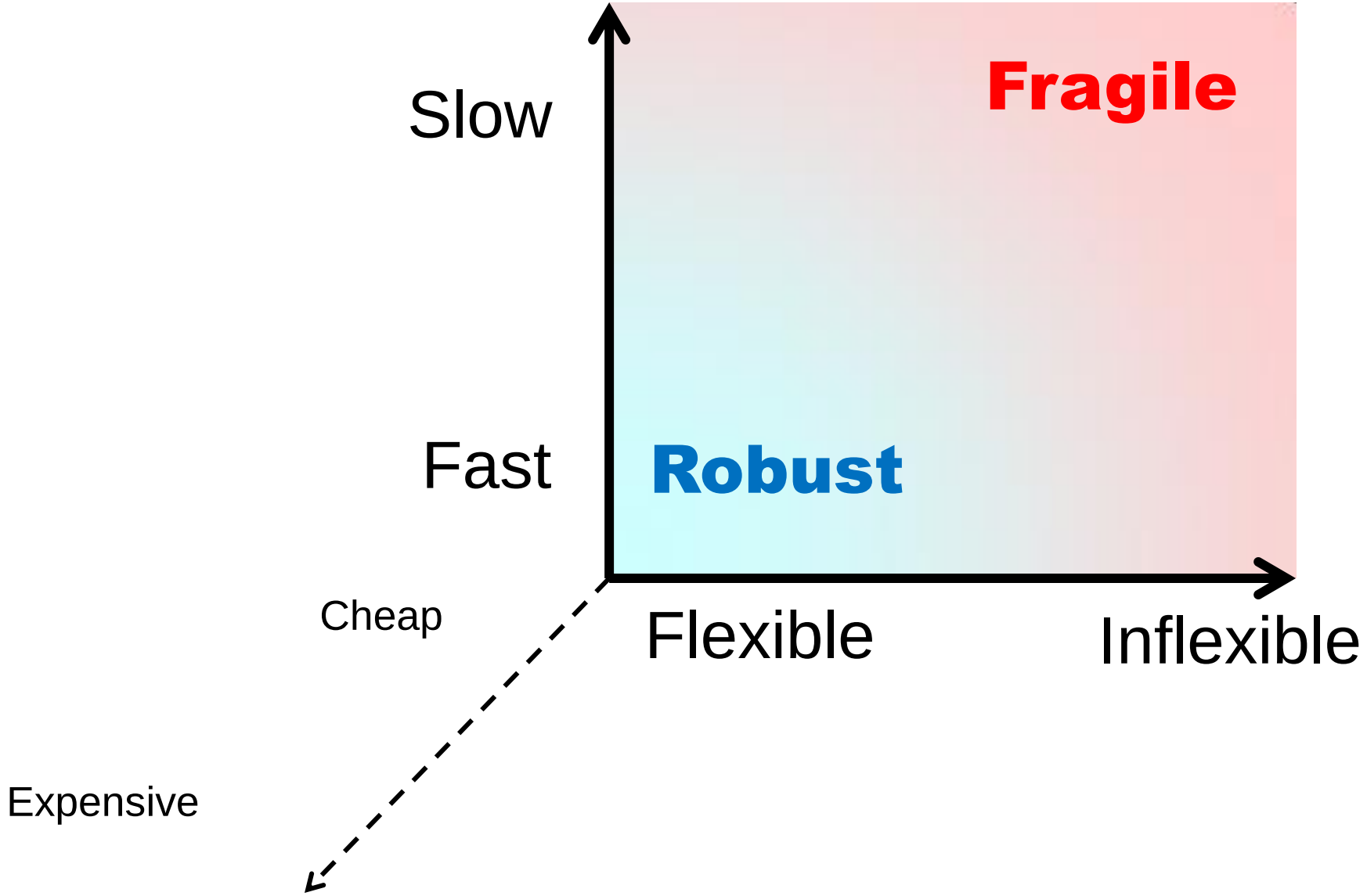
+ Neuroscience

Speed and flexibility tradeoffs

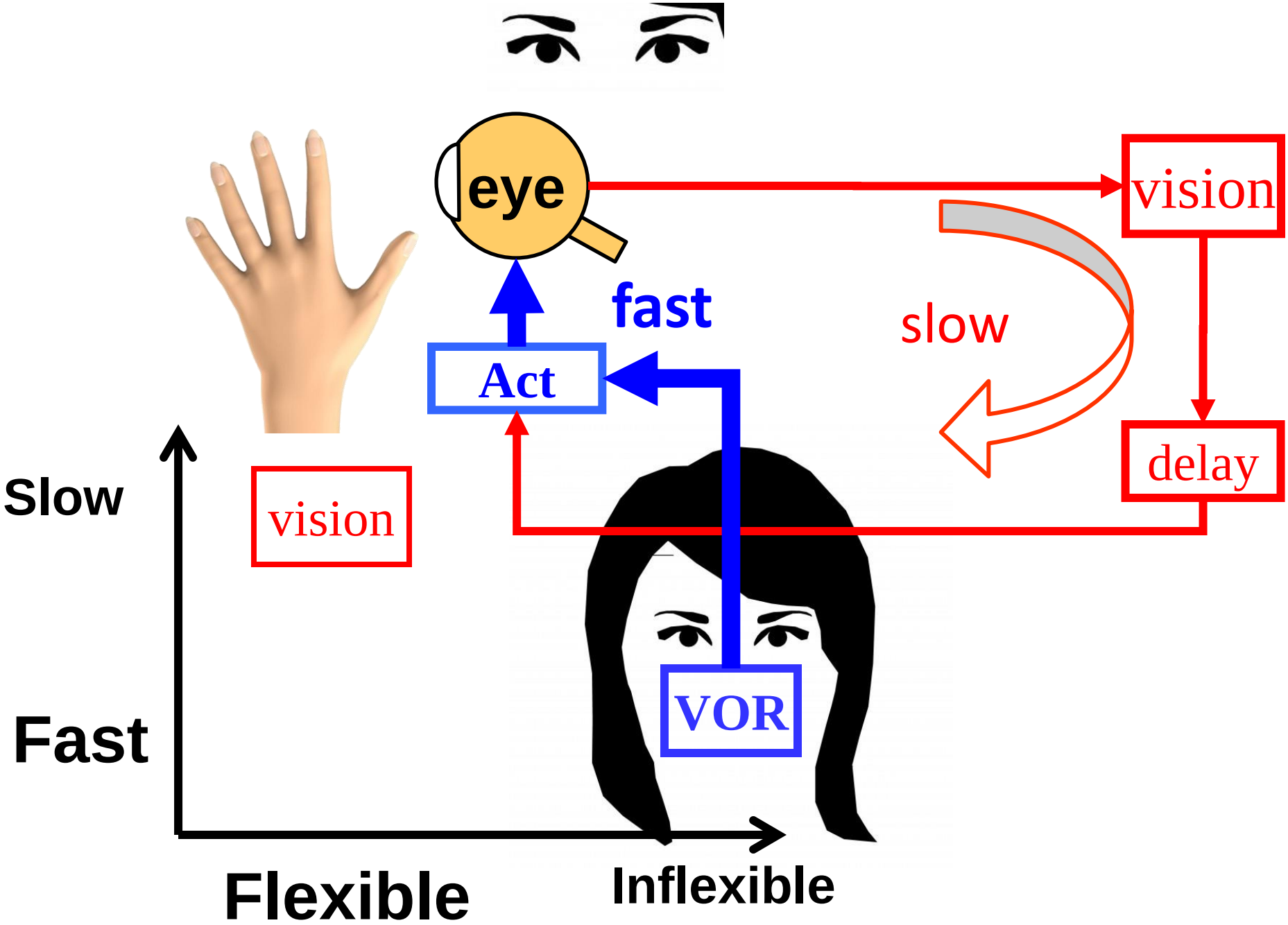
Concrete case studies?

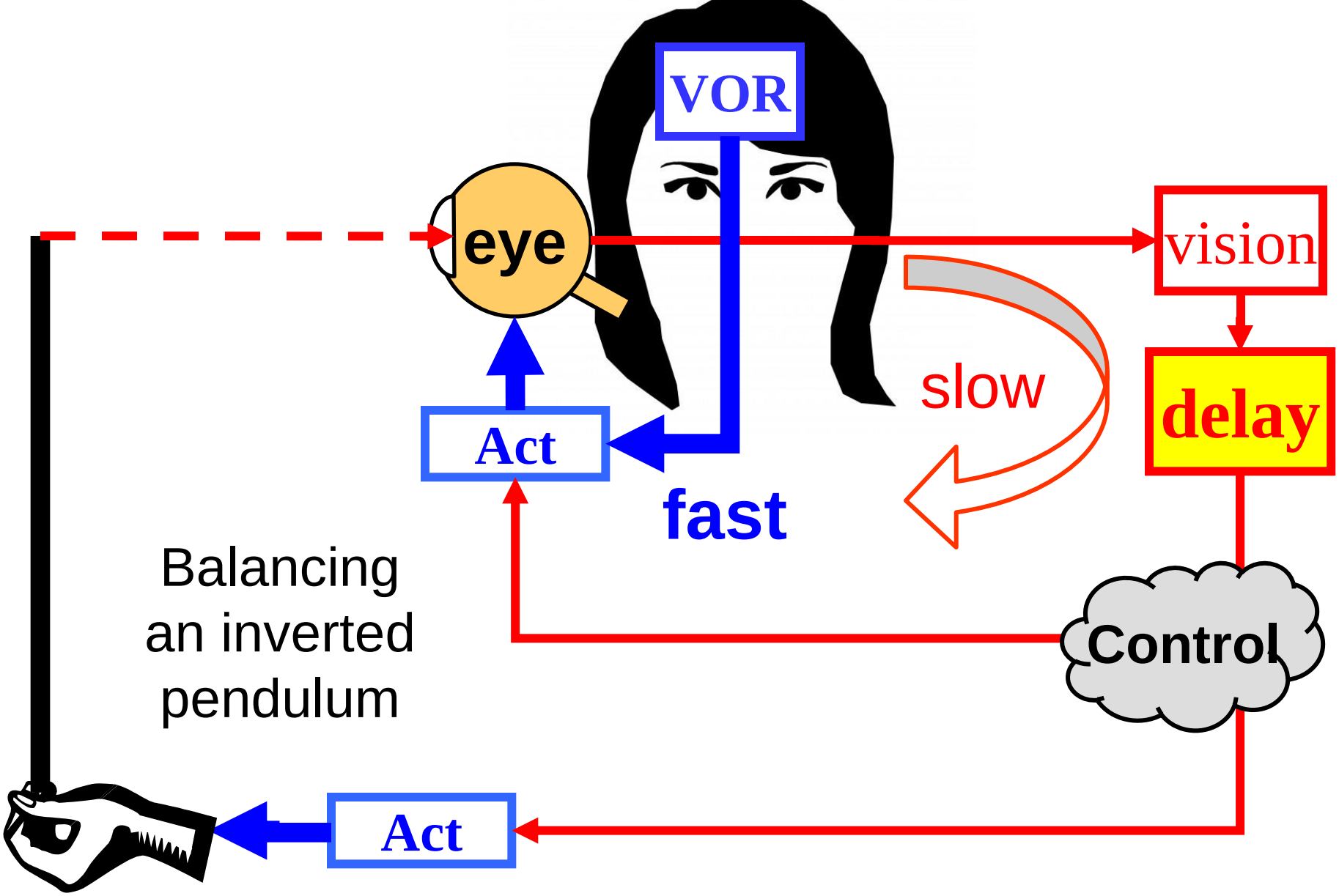
Theorems ?

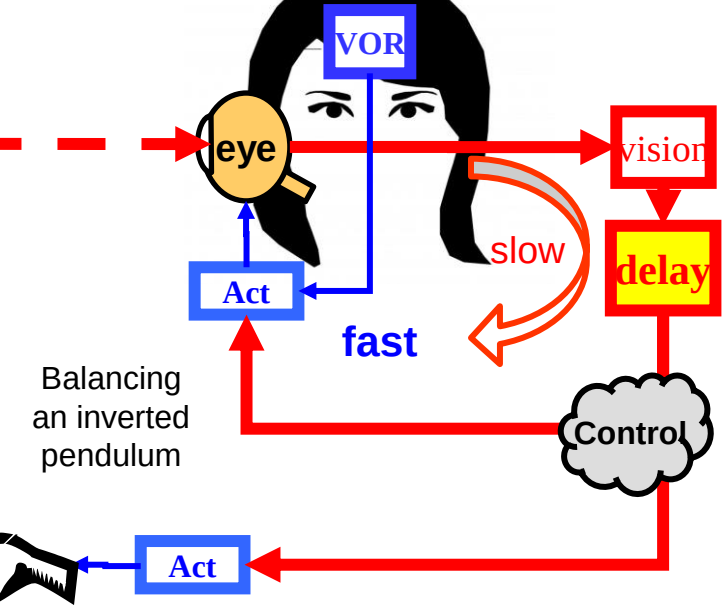


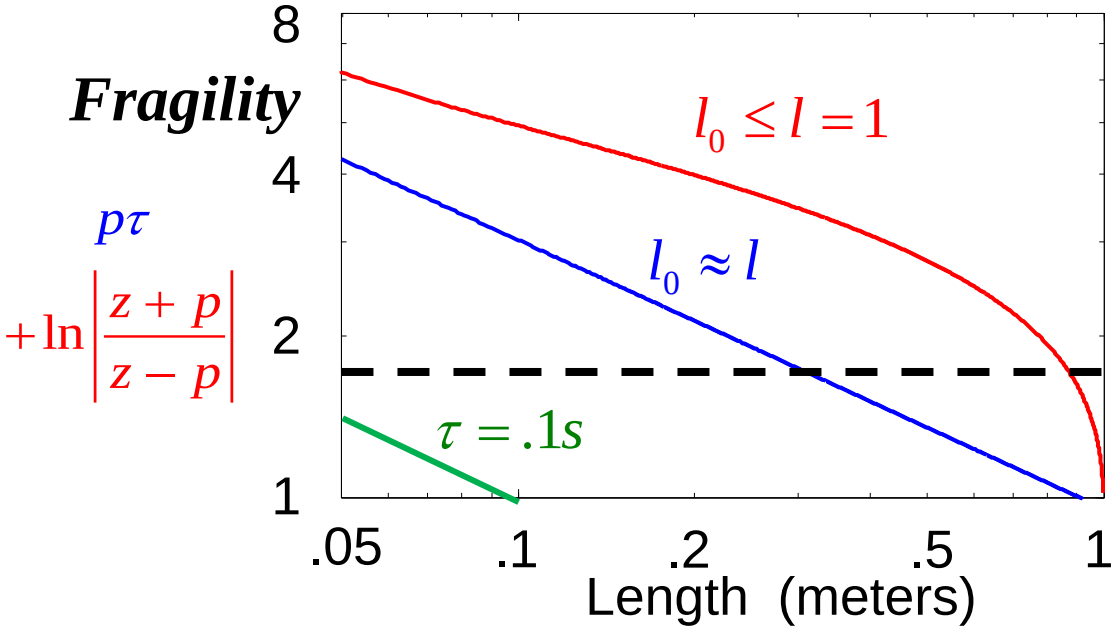


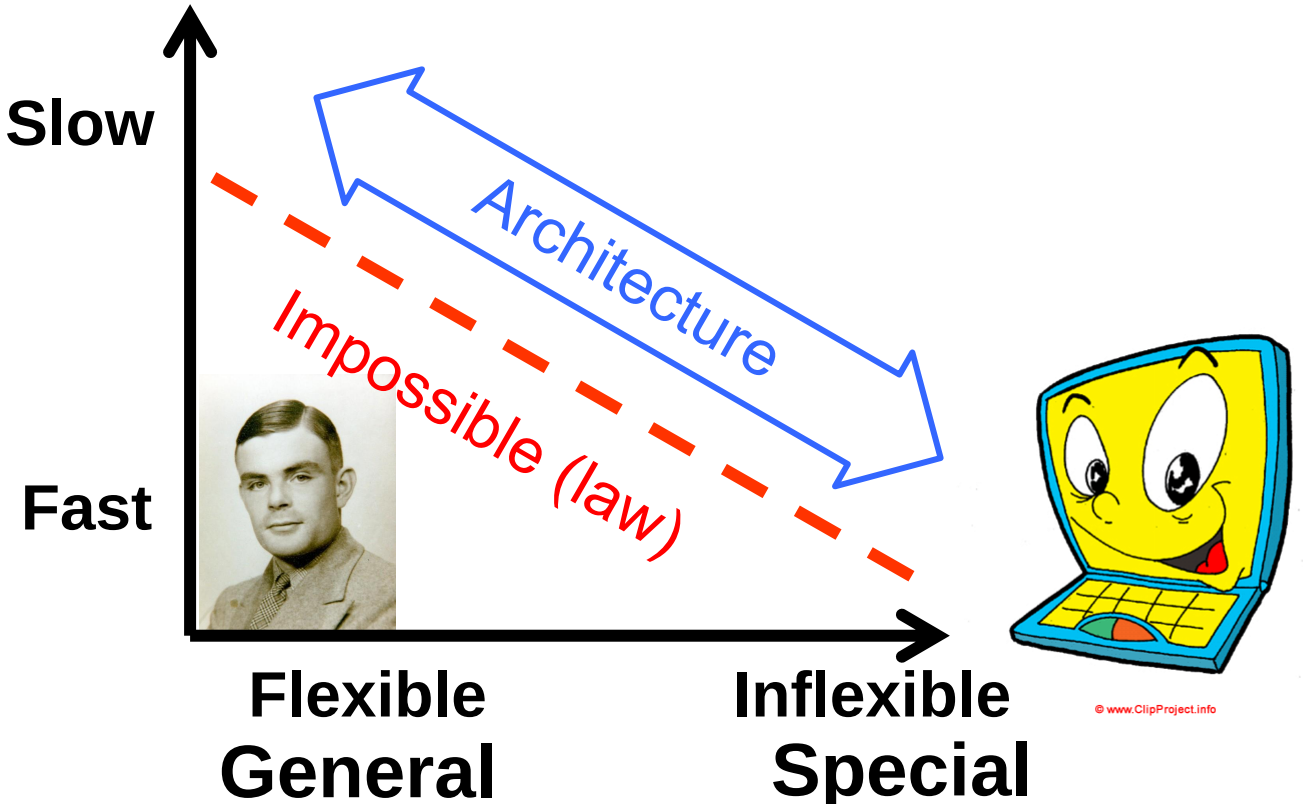
Efficient vs **Robust**

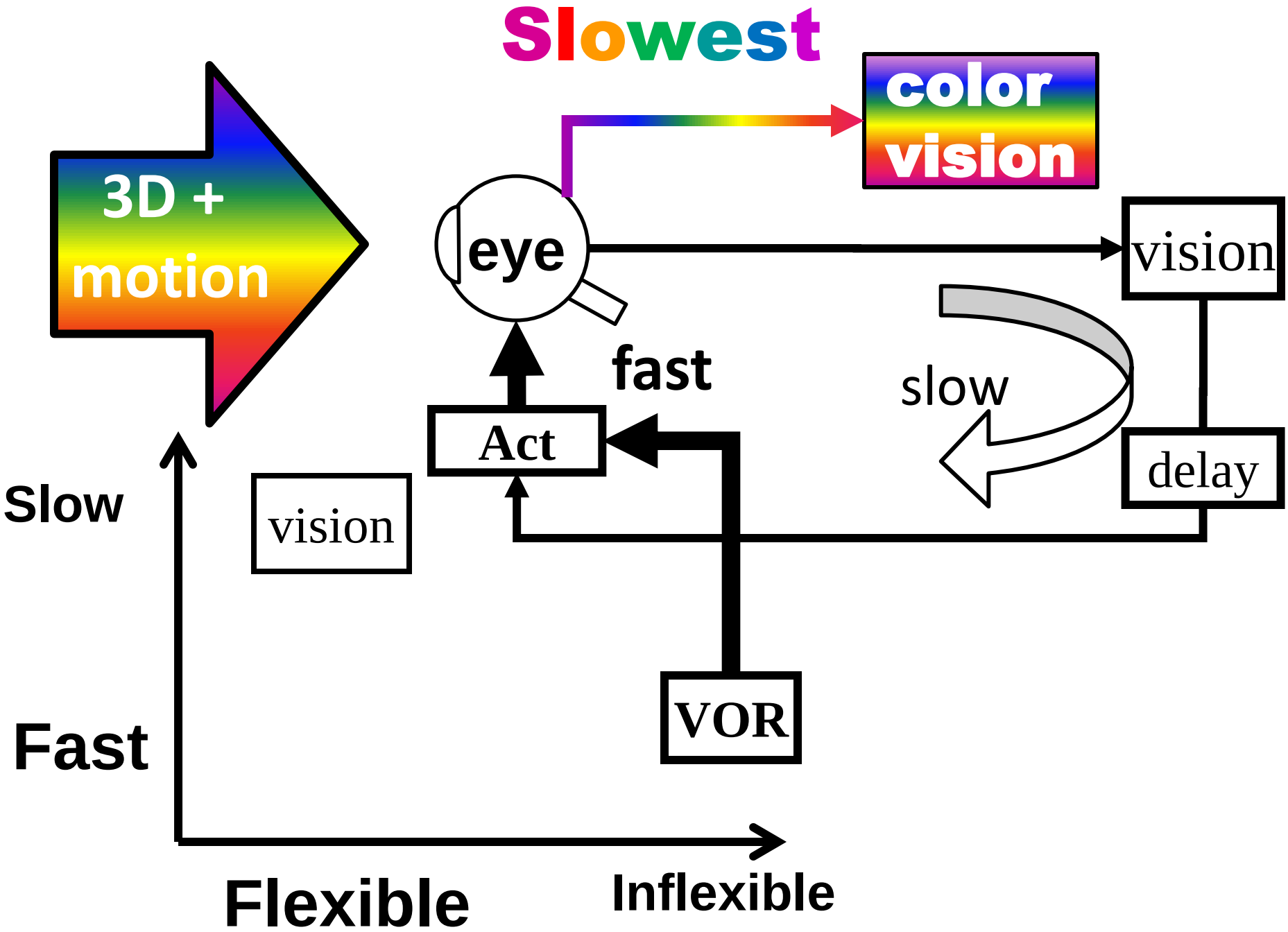

eye
vision
fast
slow
Act
delay
vision
Slow
Fast
VOR
Flexible
Inflexible


VOR
eye
vision
slow
delay
Act
fast
Balancing
an inverted
pendulum
Control
Act

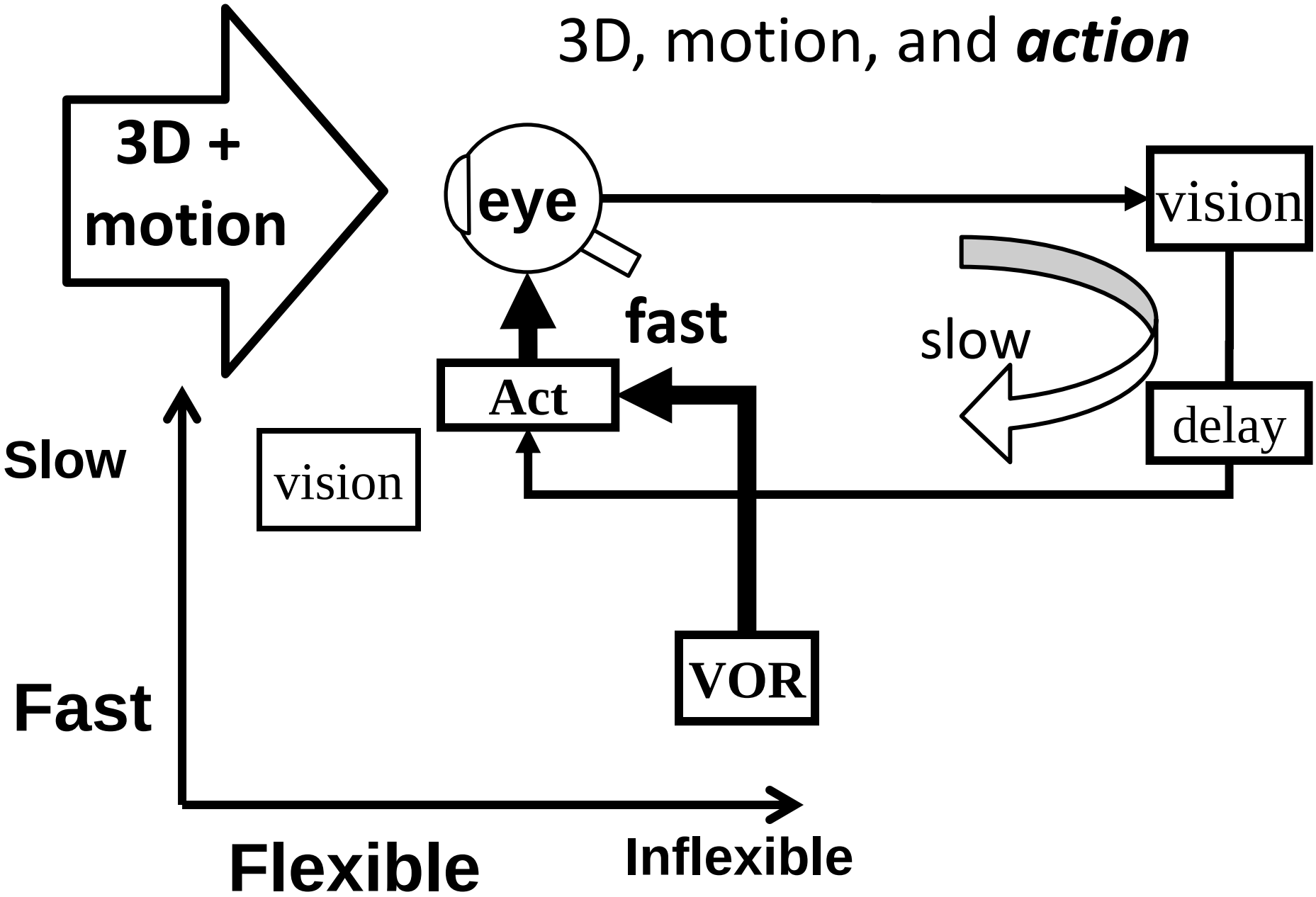

VOR
eye
vision
slow
delay
Act
fast
Balancing
an inverted
pendulum
Control
Act

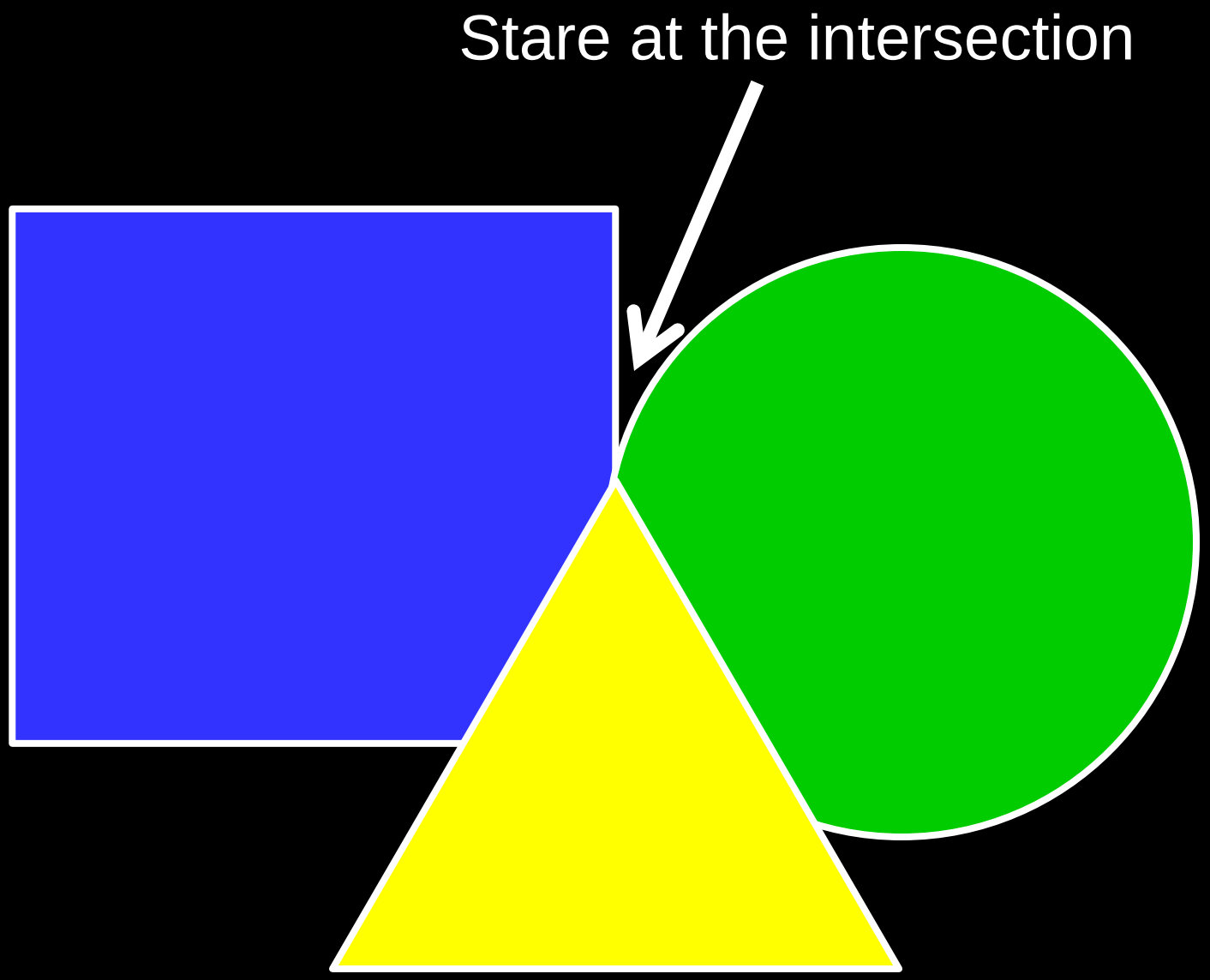

Fragility
$p\tau + \ln\left|\frac{z+p}{z-p}\right|$
8
4
2
1
$l_0 \le l = 1$
$l_0 \approx l$
$\tau = .1s$
.05
.1
.2
.5
1
Length (meters)

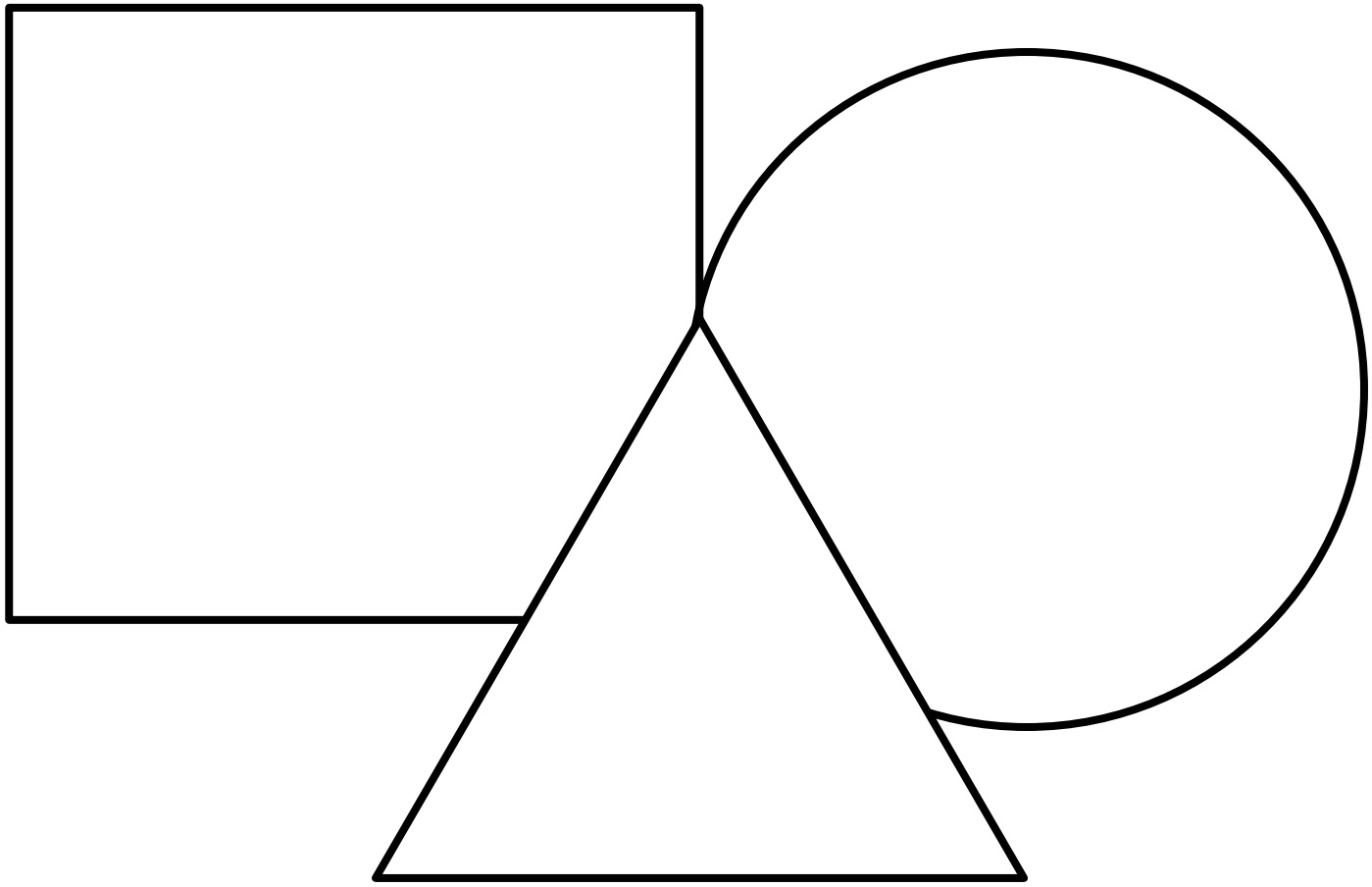

Slow
Fast
Architecture
Impossible (law)
Flexible
General
Inflexible
Special
© www.ClipProject.info

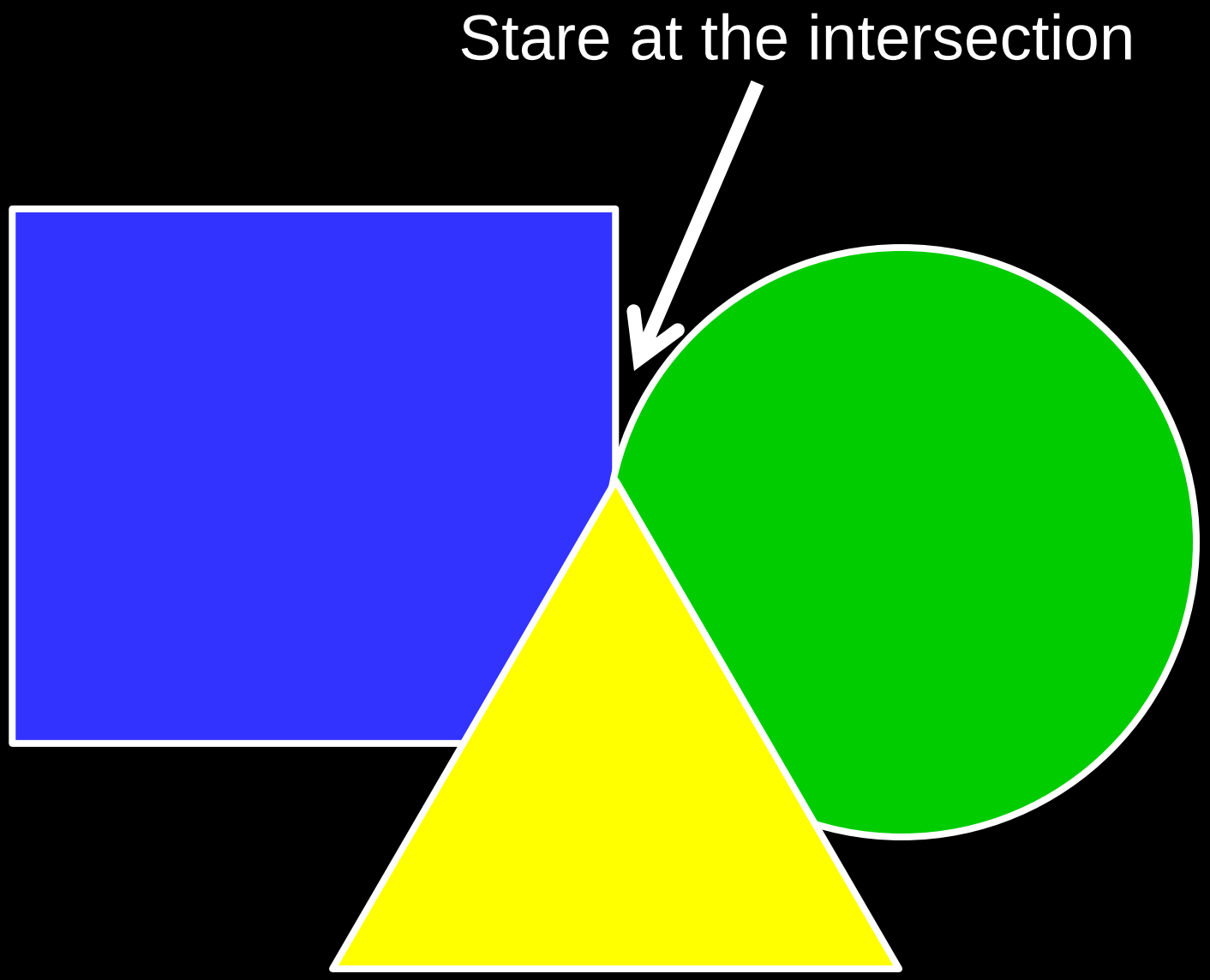

Slowest
color vision
3D + motion
eye
vision
fast
slow
Act
delay
Slow
vision
VOR
Fast
Flexible
Inflexible

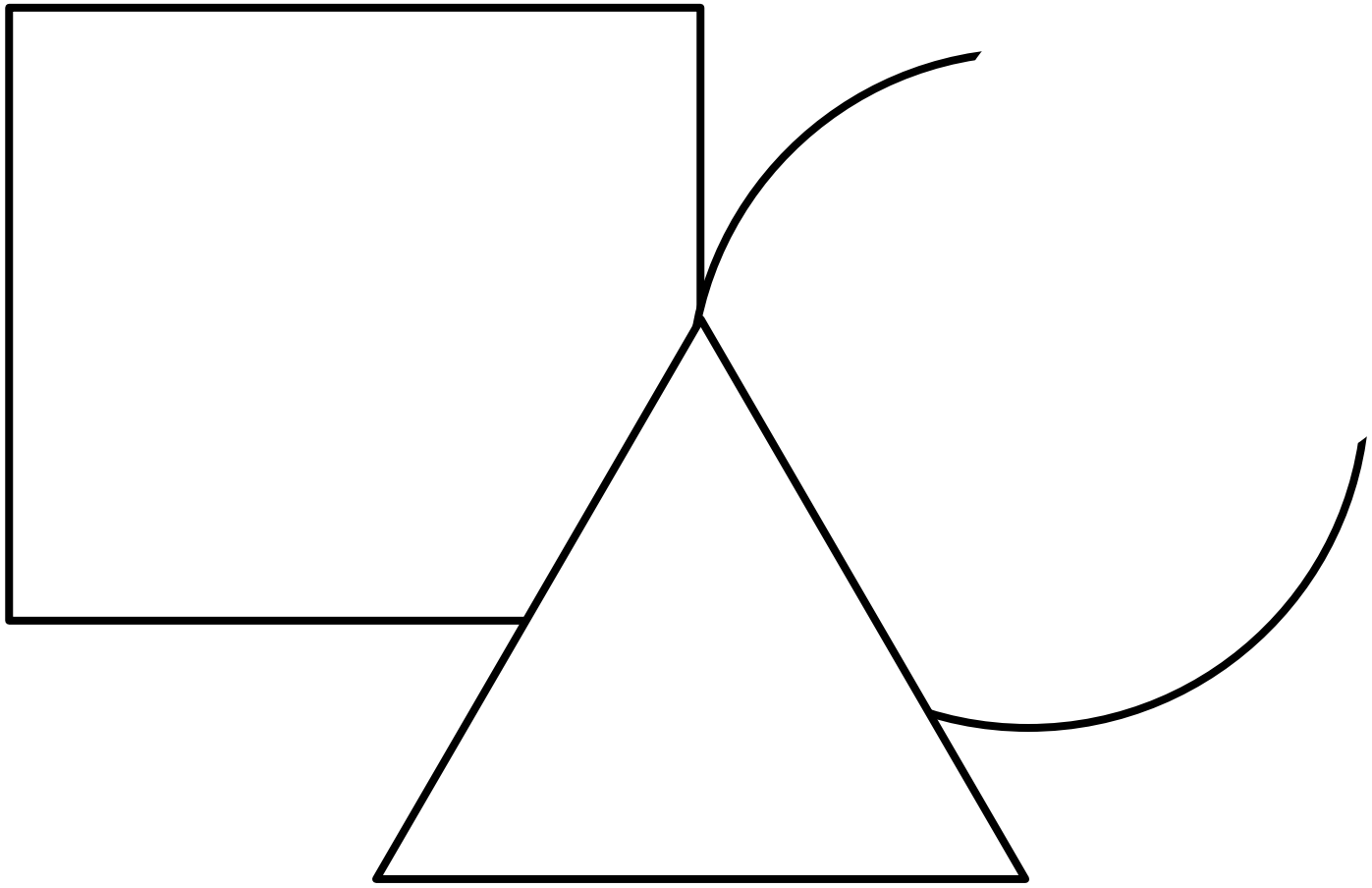
B&W (luminence only): 3D, motion, and ***action***

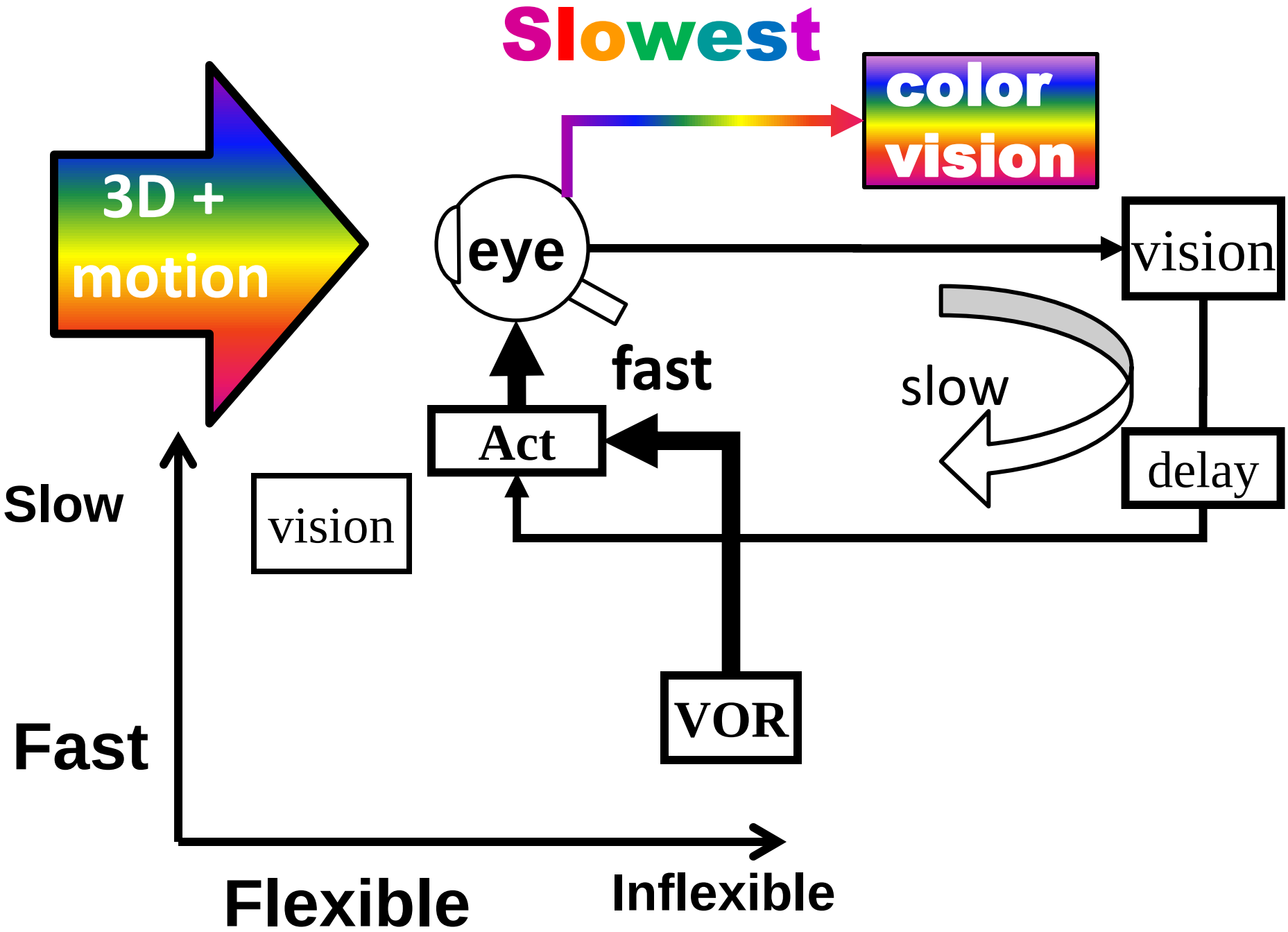



Stare at the intersection

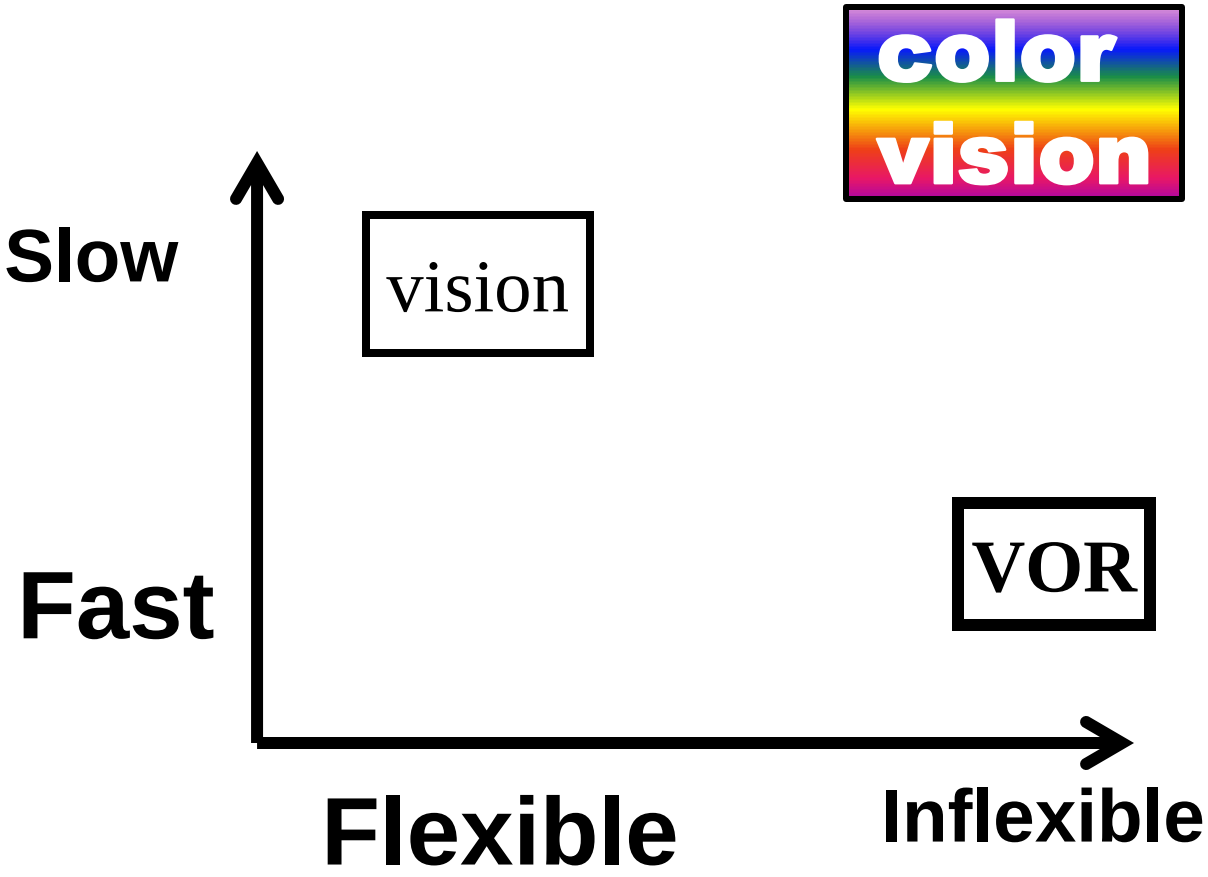


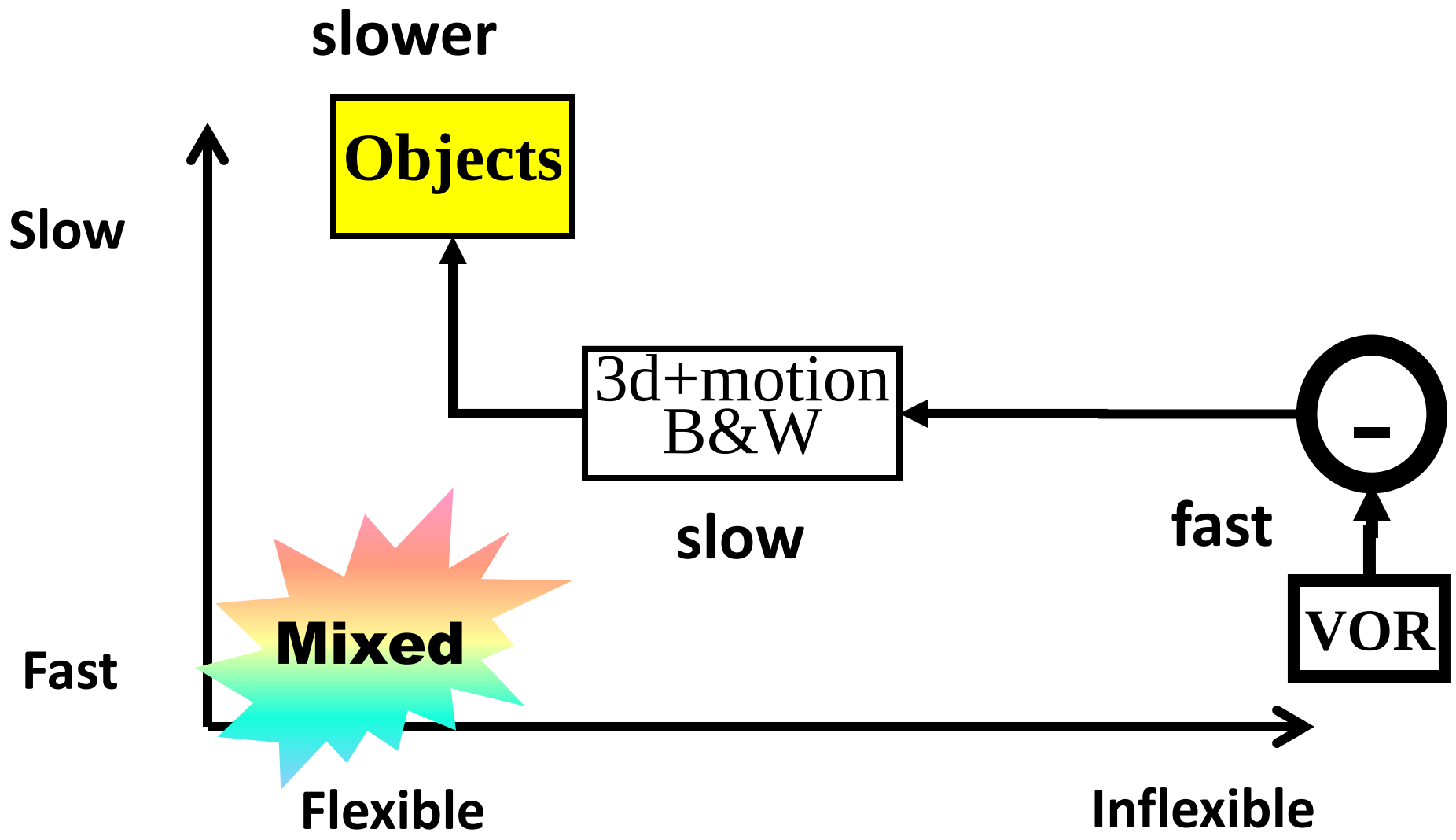

Stare at the intersection




Slowest
color vision
3D + motion
eye
vision
fast
slow
Act
delay
Slow
vision
VOR
Fast
Flexible
Inflexible

Seeing is dreaming

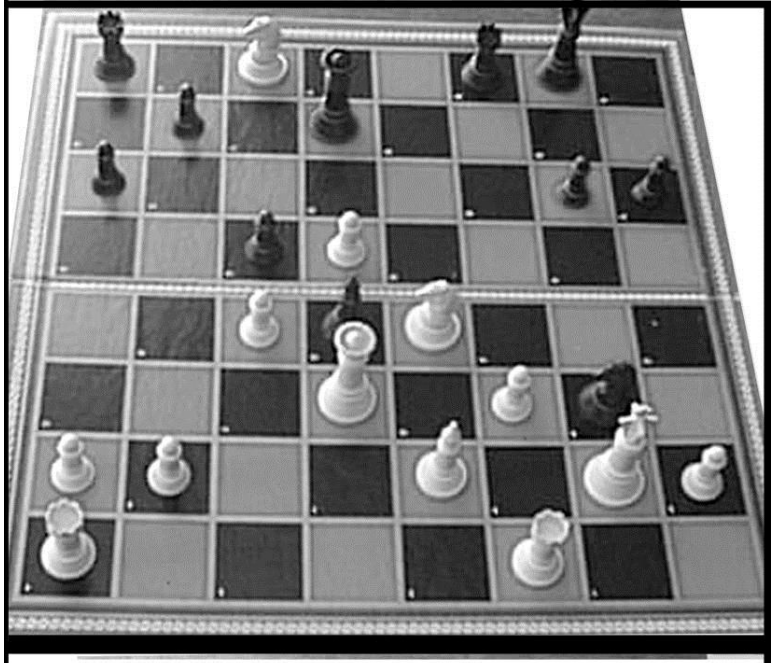


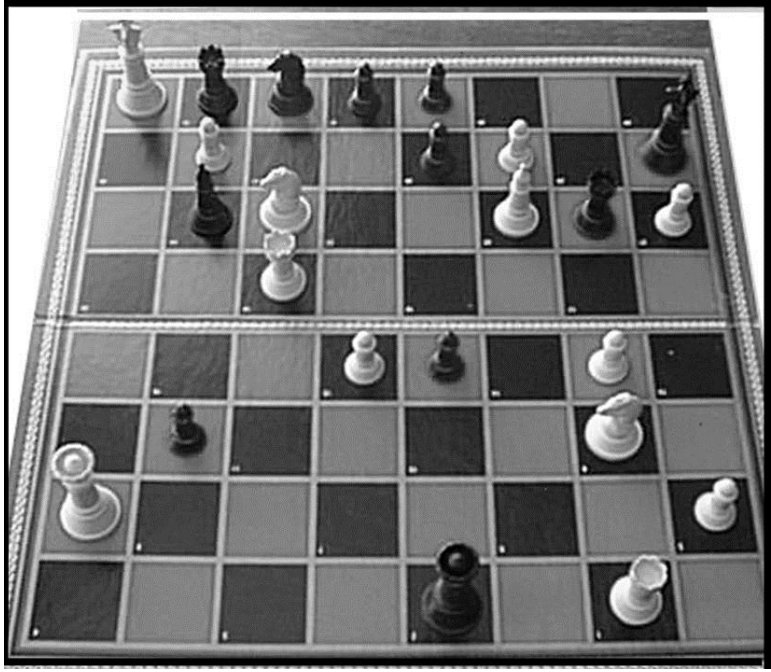

slower
Objects
Slow
3d+motion
B&W
-
slow
fast
VOR
Mixed
Fast
Flexible
Inflexible

Chess experts

- can reconstruct entire chessboard with < ~ 5s inspection
- can recognize 1e5 distinct patterns
- can play multiple games blindfolded and simultaneous
- are no better on random boards

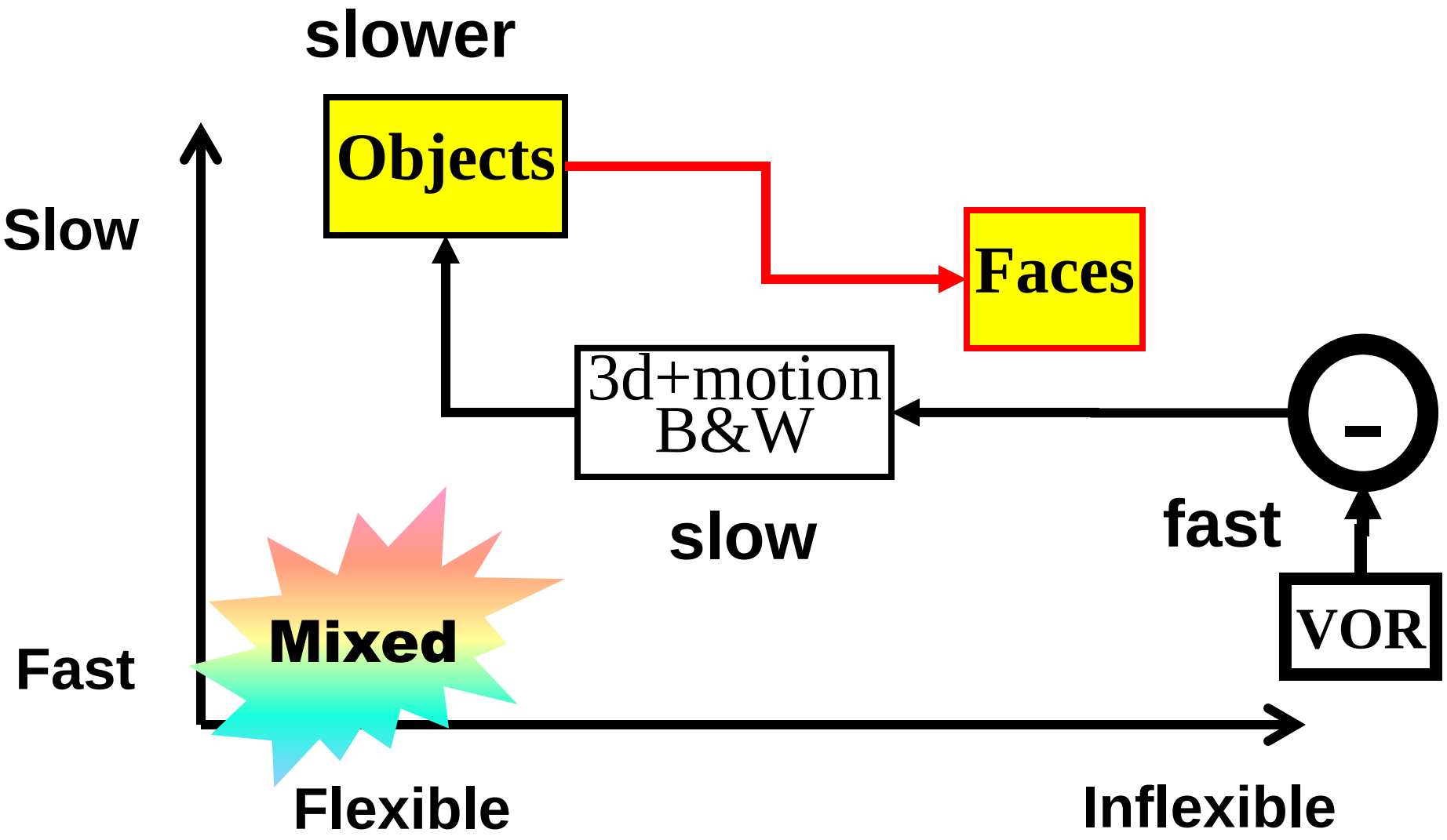
(Simon and Gilmartin, de Groot)

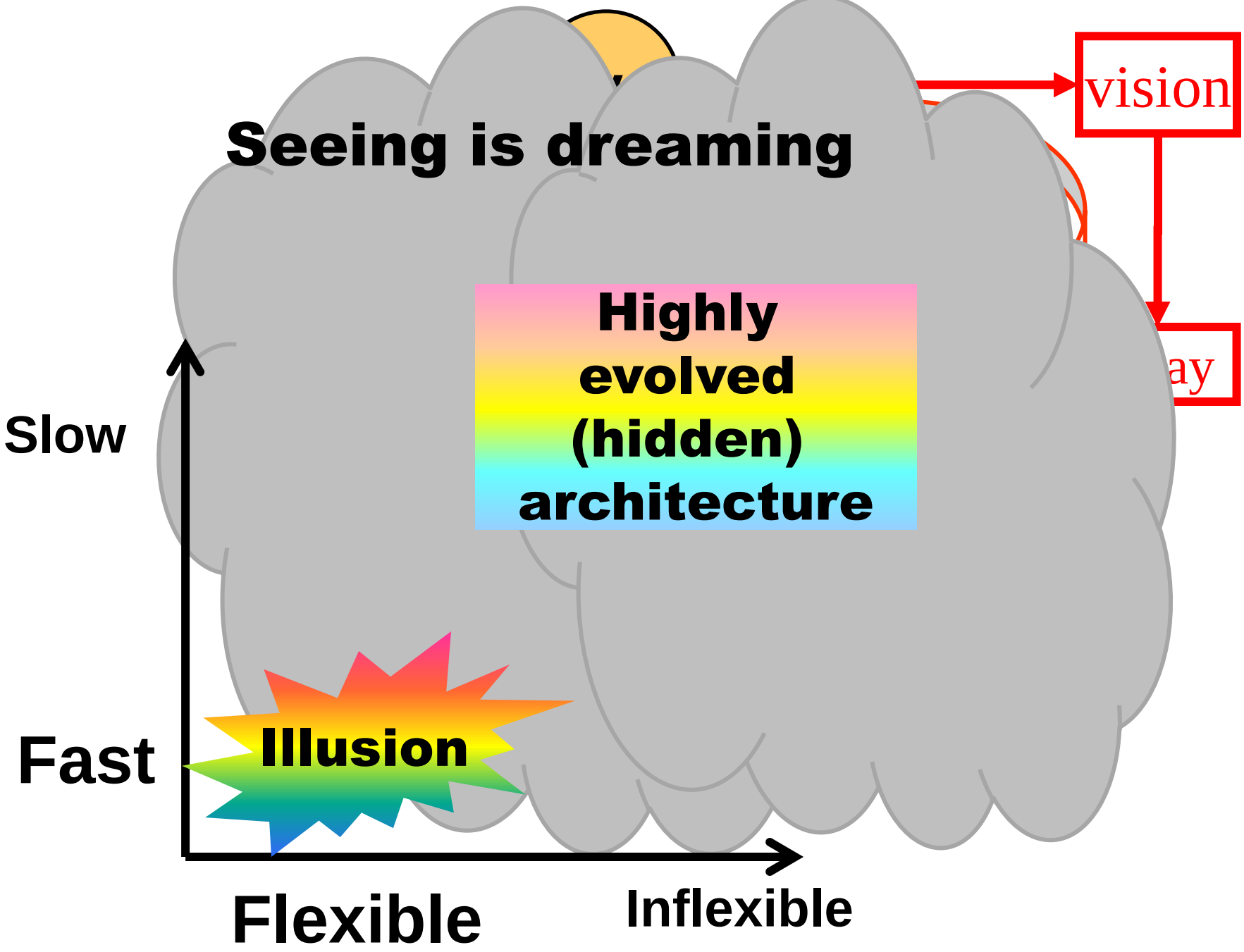




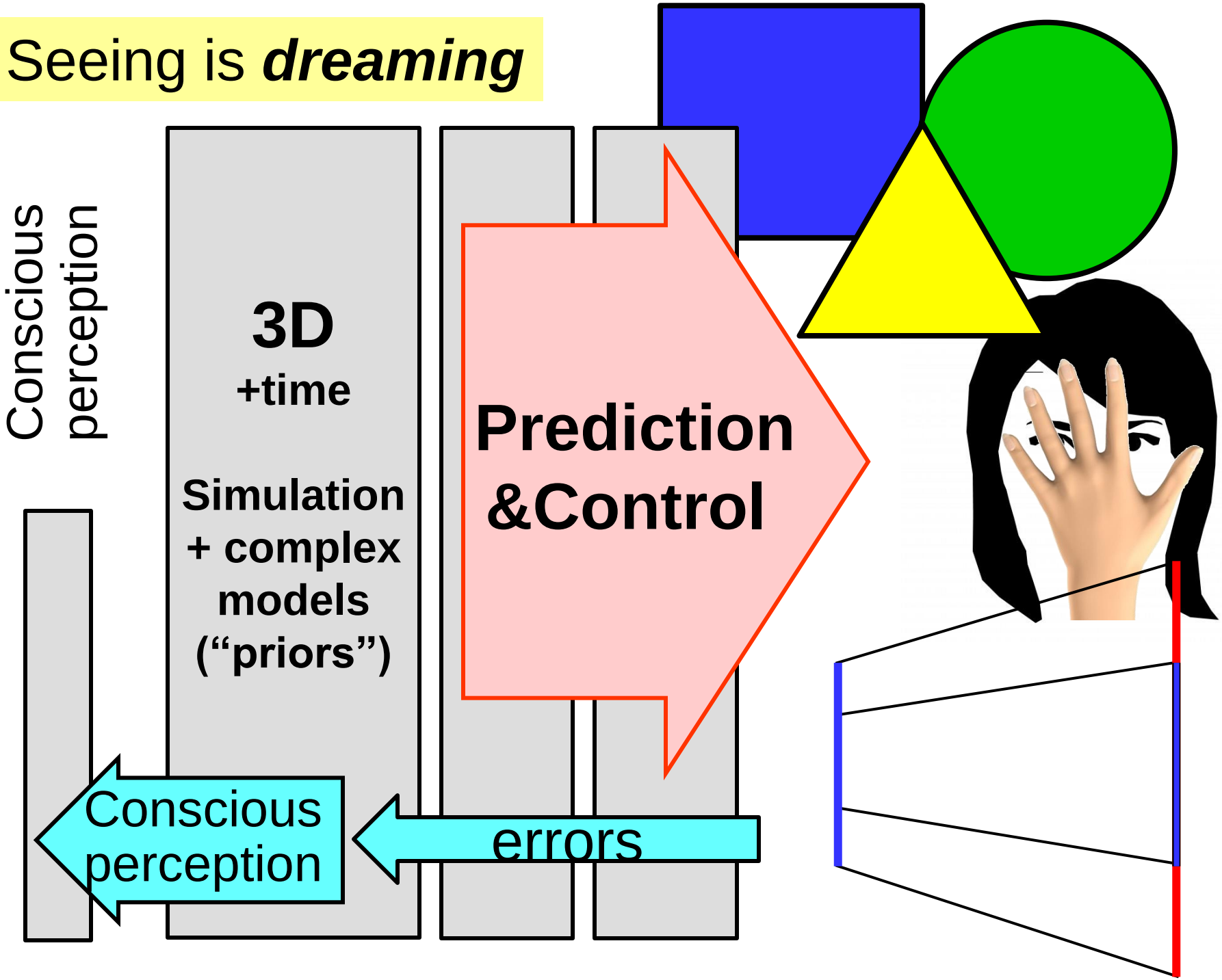
www.psywww.com/intropsych/ch07_cognition/expertise_and_domain_specific_knowledge.html

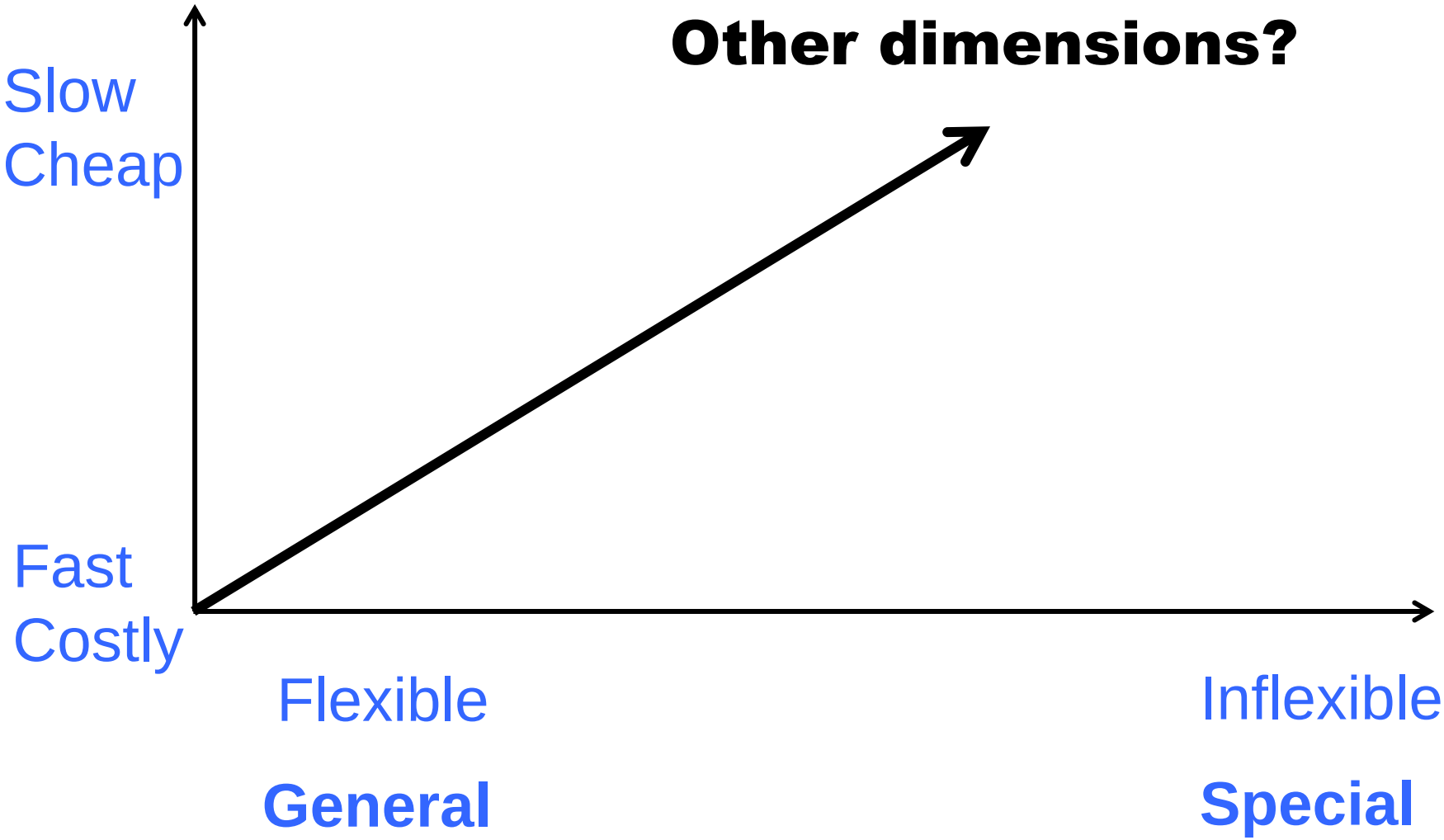
Not sure how to draw this…



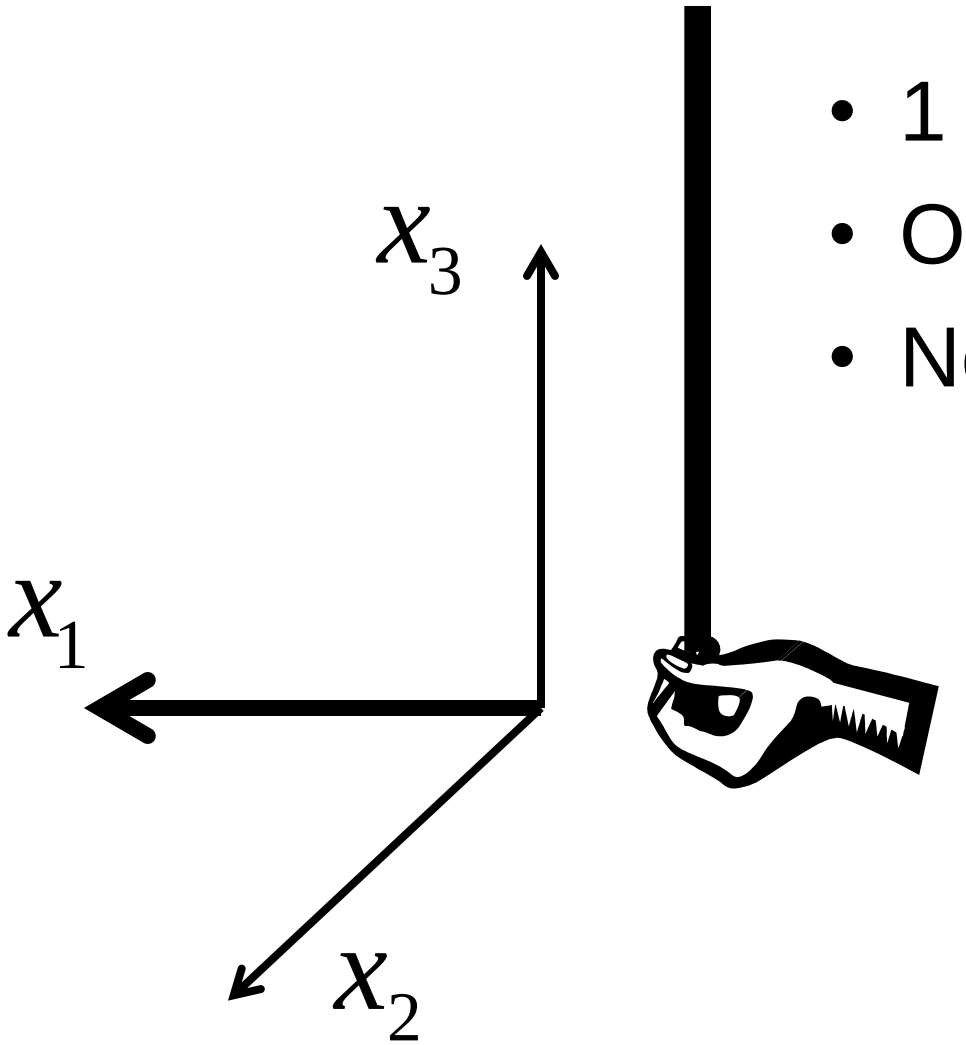

Seeing is dreaming
vision
ay
Highly evolved (hidden) architecture
Slow
Fast
Illusion
Flexible
Inflexible

Seeing is ***dreaming***

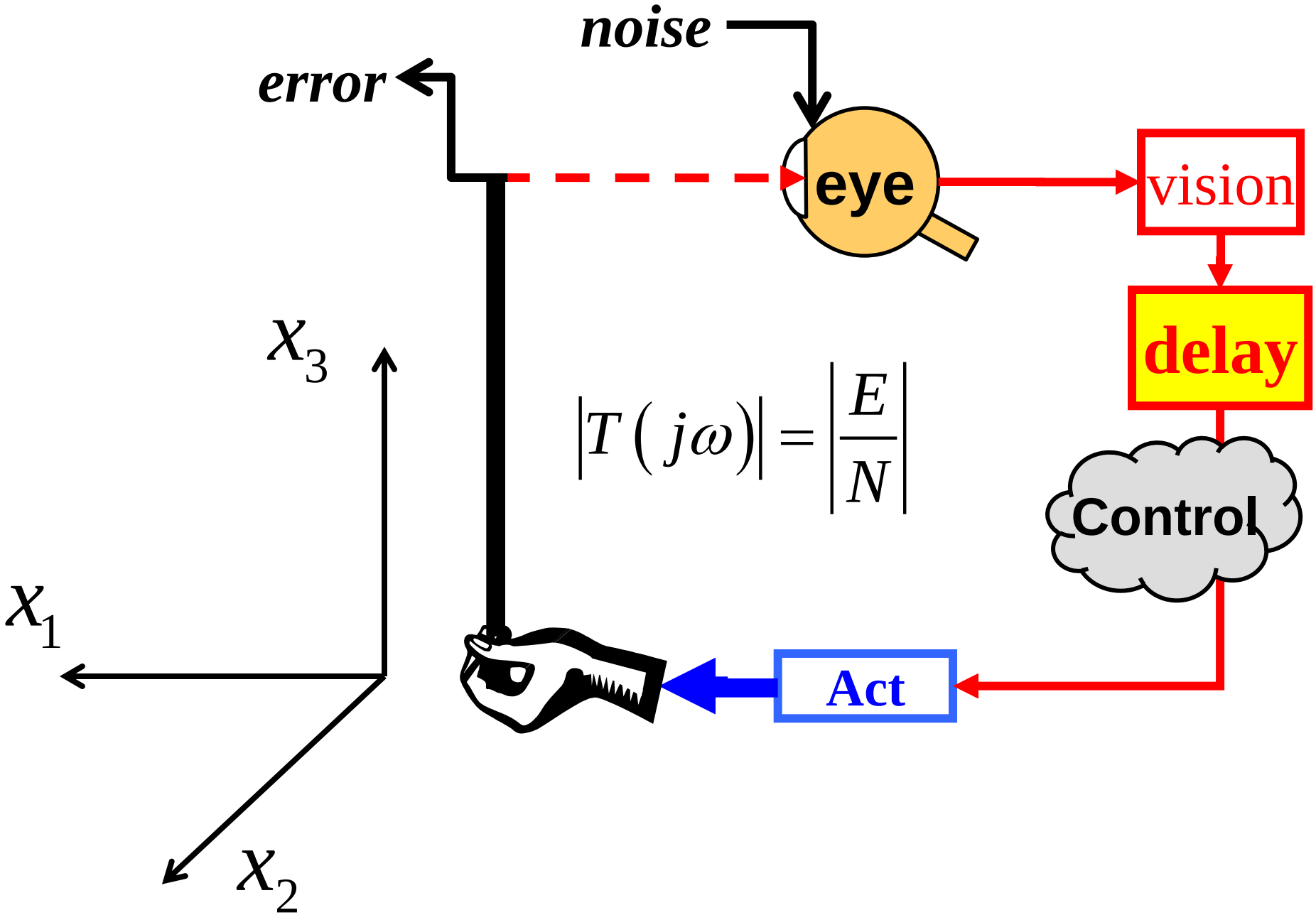


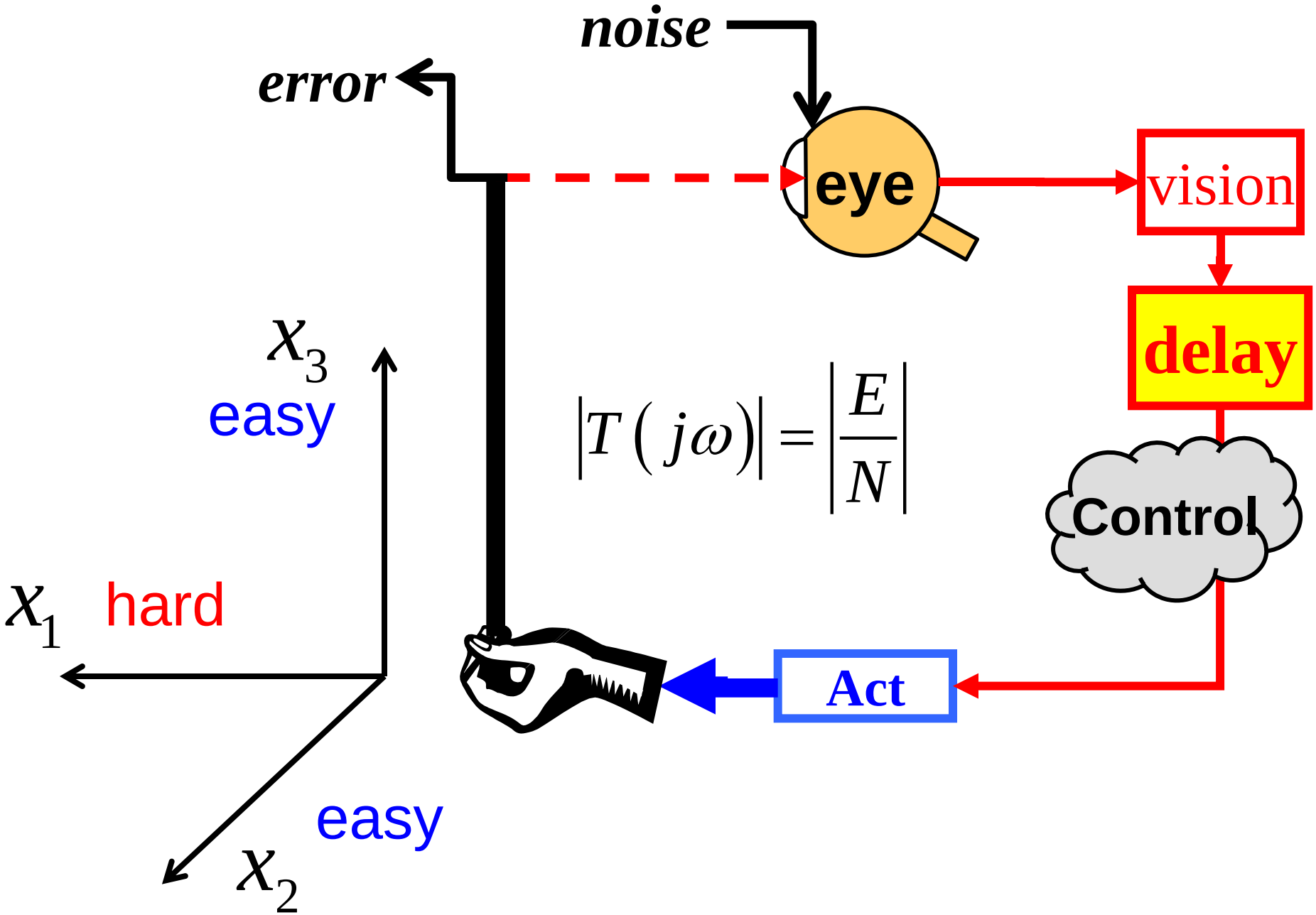

Other dimensions?
Slow
Cheap
Fast
Costly
Flexible
Inflexible
General
Special

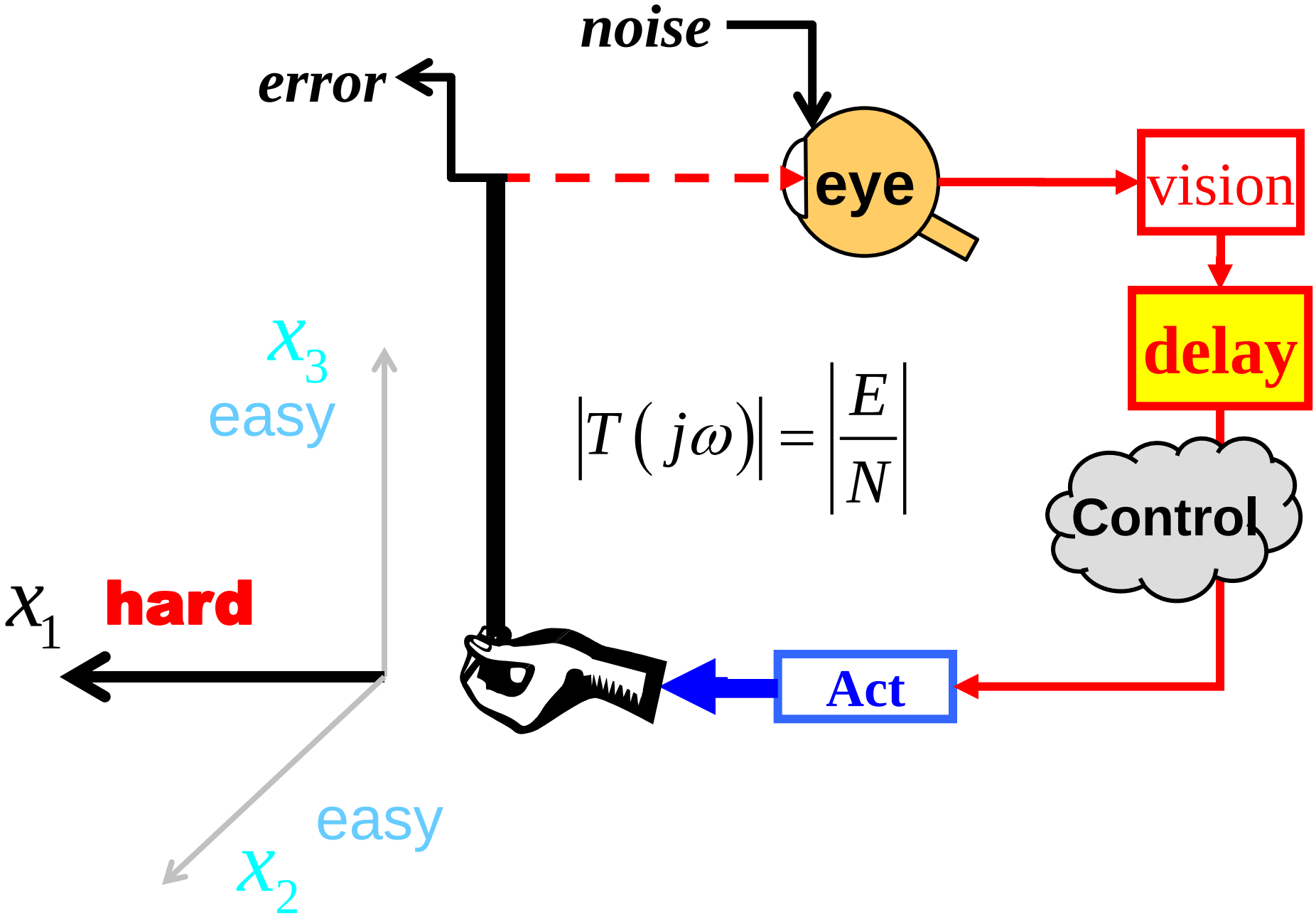
Model?



- 1 dimension, 4 states?
- Other 2 dimensions?
- New issues arise


noise
error
eye
vision
delay
Control
Act
x_3
x_1
x_2
$|T(j\omega)| = \left|\frac{E}{N}\right|$


noise
error
eye
vision
delay
Control
Act
x_3
easy
x_1
hard
x_2
easy
$|T(j\omega)| = \left|\frac{E}{N}\right|$


noise
error
eye
vision
delay
Control
Act
$|T(j\omega)| = \left|\frac{E}{N}\right|$
x_3
easy
x_1
hard
easy
x_2

Main lessons

- Theory: hard limits on closed loop performance, aggravated by
  - Instability (unstable poles)
  - Delay
  - Unstable zeros
- Neuroscience specific

Instabilities in technology

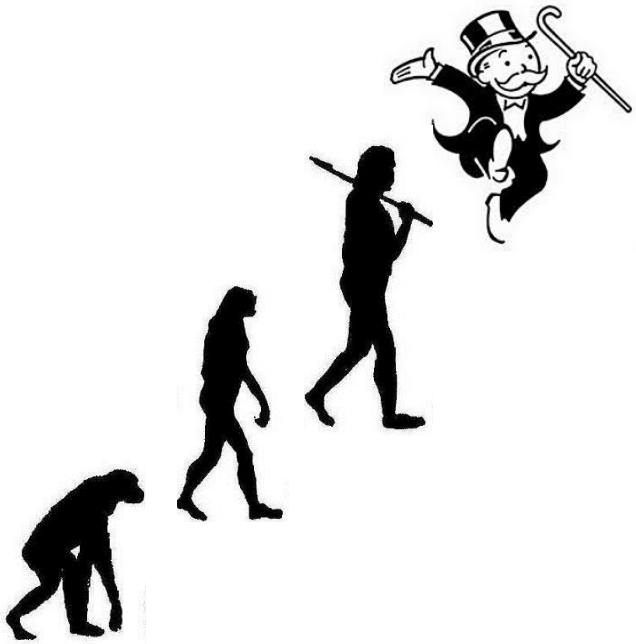
- Efficiency
- Autocatalysis

- The future

Select instabilities in biology

Working backwards

- Money/finance/lobbyists/etc
- Society/agriculture/weapons/etc
- Bipedalism
- Maternal care
- Warm blood
- Flight
- Mitochondria
- Translation (ribosomes)
- Glycolysis (see 2011 Science paper)



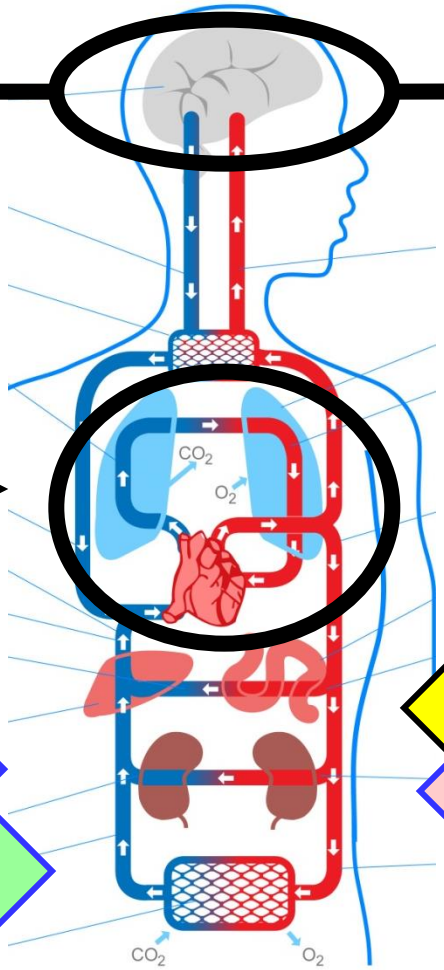
Homeostasis

controls

heart rate
ventilation
vasodilation
coagulation
inflammation
digestion
storage
…

errors

O2
BP
pH
Glucose
Energy store
Blood volume
…

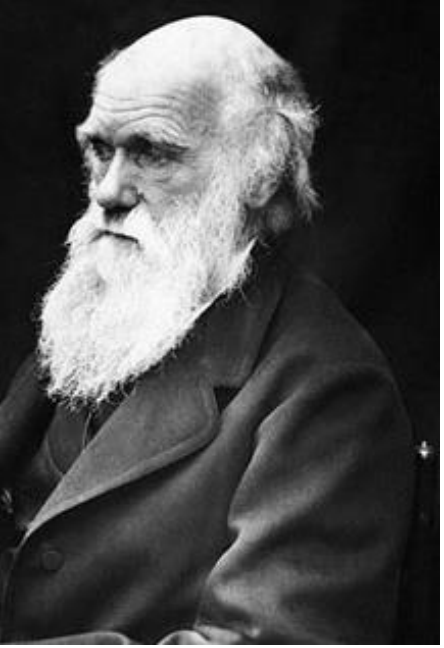


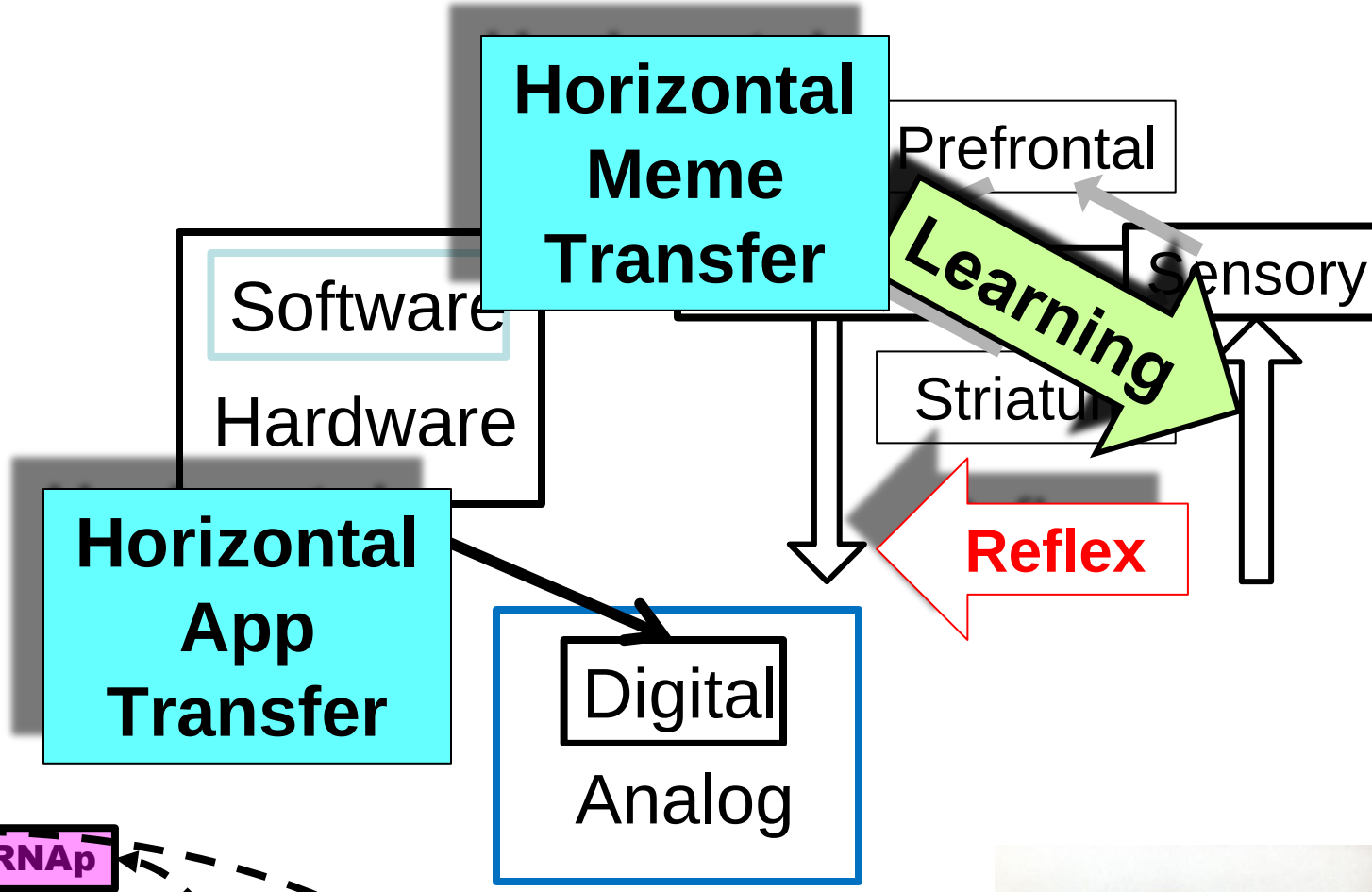
energy
trauma
infection

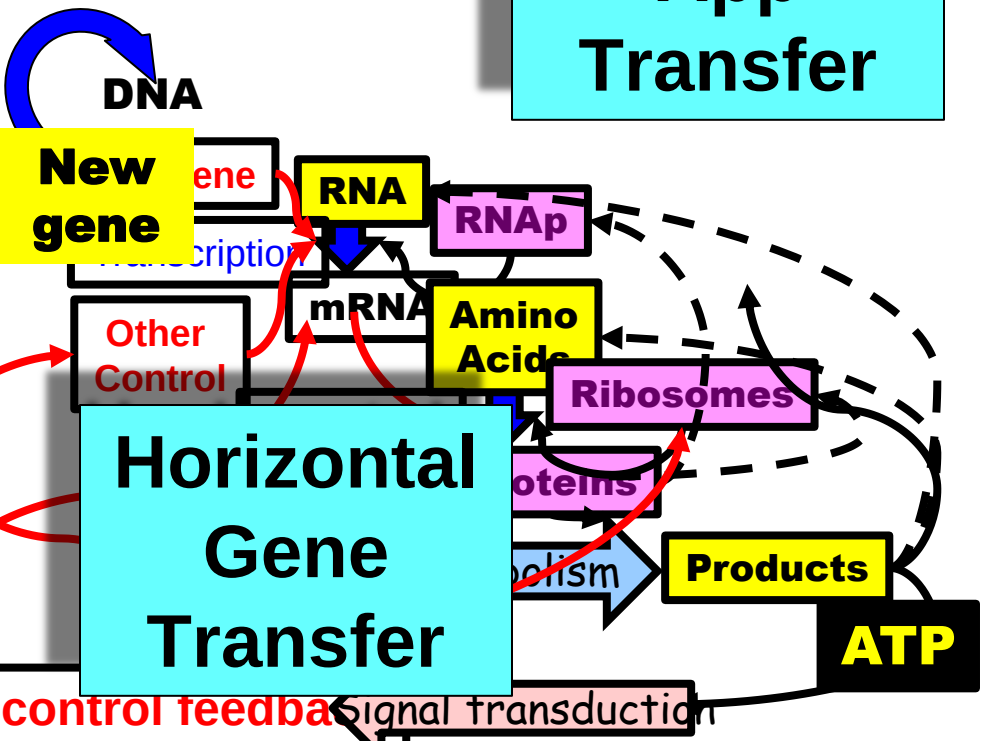
breath
heart beat
sensor

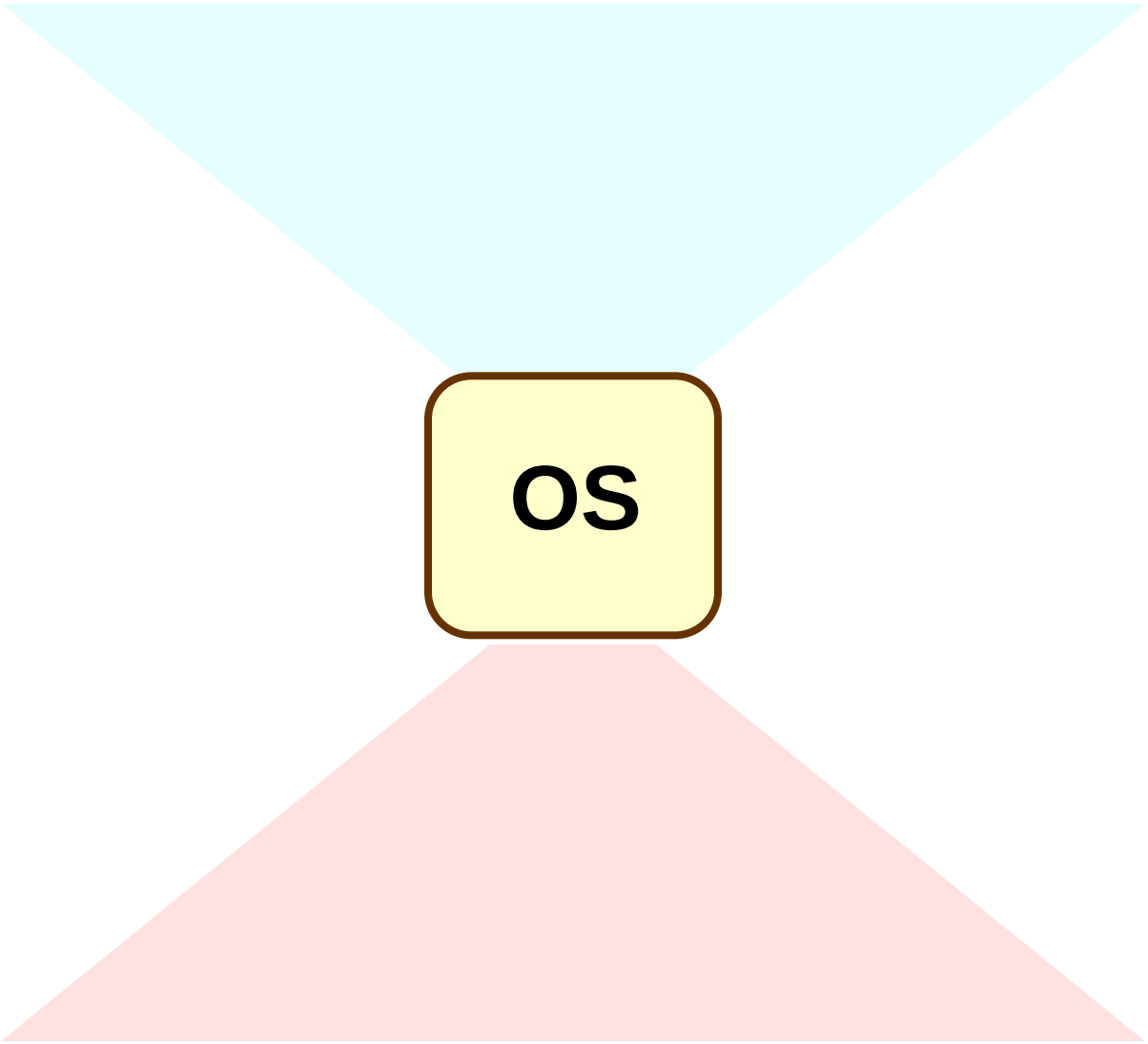
external disturbances

internal noise

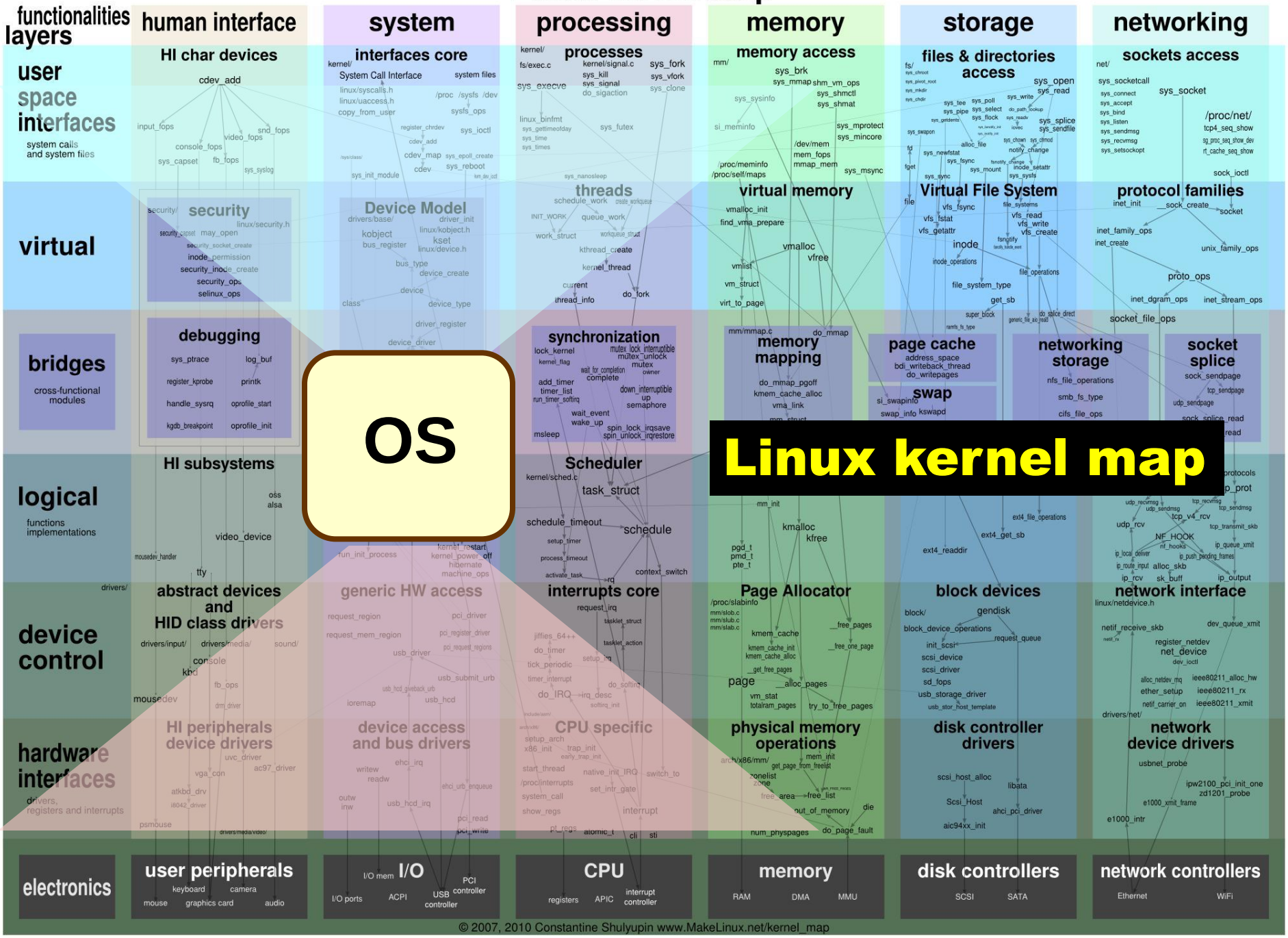



Horizontal Meme Transfer
Prefrontal
Learning
Sensory
Software
Hardware
Striatum
Reflex
Horizontal App Transfer
Digital
Analog


DNA
New gene
RNA
RNAp
mRNA
Other Control
Amino Acids
Ribosomes
Products
ATP
Horizontal Gene Transfer
control feedba
Signal transduction


OS

Linux kernel map



© 2007, 2010 Constantine Shulyupin www.MakeLinux.net/kernel_map

Linux kernel map

2.6.32

functionalities
layers

human interface | system | processing | memory | storage | networking

user space interfaces
system calls and system files

virtual

bridges
cross-functional modules

logical
functions implementations

device control

hardware interfaces
drivers, registers and interrupts

electronics

HI char devices
cdev_add
input_fops
snd_fops
video_fops
console_fops
sys_capset
fb_fops
sys_syslog

interfaces core
kernel/
System Call Interface
system files
linux/syscalls.h
linux/uaccess.h
copy_from_user
/proc /sysfs /dev
sysfs_ops
register_chrdev
sys_ioctl
cdev_add
cdev_map
sys_epoll_create
cdev
sys_reboot
sys_init_module

processes
kernel/
fs/exec.c
kernel/signal.c
sys_fork
sys_kill
sys_vfork
sys_execve
sys_signal
sys_clone
do_sigaction
linux_binfmt
sys_gettimeofday
sys_futex
sys_time
sys_times

memory access
mm/
sys_brk
sys_mmap
shm_vm_ops
sys_shmctl
sys_sysinfo
sys_shmat
si_meminfo
sys_mprotect
sys_mincore
/dev/mem
mem_fops
mmap_mem
/proc/meminfo

files & directories access
fs/
sys_open
sys_read
sys_write
sys_tee
sys_poll
sys_pipe
sys_select
sys_flock
sys_splice
sys_sendfile
sys_getdents
sys_swapon
alloc_file
sys_chown
sys_chmod
sys_newfstat
notify_change
sys_fsync
sys_mount
inode_setattr
sys_sysfs

sockets access
net/
sys_socketcall
sys_connect
sys_accept
sys_bind
sys_listen
sys_sendmsg
sys_recvmsg
sys_setsockopt
sys_socket
/proc/net/
tcp4_seq_show
sock_ioctl

security
security/
security
linux/security.h
security_capset
may_open
security_socket_create
inode_permission
security_inode_create
security_ops
selinux_ops

Device Model
drivers/base/
driver_init
kobject
linux/kobject.h
bus_register
kset
linux/device.h
bus_type
device_create
device
class
device_type
driver_register
device_driver
probe
load_module
module
module_param
kernel_param
kobject_uevent_init
kobject_uevent

threads
schedule_work
create_workqueue
INIT_WORK
queue_work
work_struct
workqueue_struct
kthread_create
kernel_thread
current
do_fork
thread_info

virtual memory
vmalloc_init
find_vma_prepare
vmalloc
vfree
vmlist
vm_struct
virt_to_page

Virtual File System
file
vfs_fstat
vfs_getattr
vfs_read
vfs_write
vfs_create
file_systems
file_operations
file_system_type
get_sb

protocol families
inet_init
__sock_create
socket
inet_family_ops
inet_create
unix_family_ops
proto_ops
inet_dgram_ops
inet_stream_ops
socket_file_ops

debugging
sys_ptrace
log_buf
register_kprobe
printk
handle_sysrq
oprofile_start
kgdb_breakpoint
oprofile_init

synchronization
lock_kernel
mutex_lock_interruptible
mutex_unlock
kernel_flag
mutex
wait_for_completion
owner
complete
add_timer
down_interruptible
timer_list
up
run_timer_softirq
semaphore
wait_event
wake_up
msleep
spin_lock_irqsave
spin_unlock_irqrestore

memory mapping
mm/mmap.c
do_mmap
do_mmap_pgoff
kmem_cache_alloc
vma_link
mm_struct
vm_area_struct

page cache
address_space
bdi_writeback_thread
do_writepages

swap
si_swapinfo
swap_info
kswapd
do_swap_page
wakeup_kswapd

networking storage
nfs_file_operations
smb_fs_type
cifs_file_ops
iscsi_tcp_transport

socket splice
sock_sendpage
tcp_sendpage
udp_sendpage
sock_splice_read
tcp_splice_read

HI subsystems
oss
alsa
video_device
mousedev_handler
tty

system run
init/
kernel/
boot, shutdown
power management
init/main.c
start_kernel
mount_root
do_initcalls
run_init_process
kernel_restart
kernel_power_off
hibernate
machine_ops

Scheduler
kernel/sched.c
task_struct
schedule_timeout
schedule
setup_timer
process_timeout
activate_task
context_switch
rq

logical memory
physically mapped memory
kmalloc
kfree
pgd_t
pmd_t
pte_t

logical file systems
ext4_file_operations
ext4_get_sb
ext4_readdir

protocols
proto
/proc/net/protocols
udp_prot
tcp_prot
udp_recvmsg
tcp_recvmsg
udp_sendmsg
tcp_sendmsg
tcp_v4_rcv
udp_rcv
tcp_transmit_skb
NF_HOOK
nf_hooks
ip_queue_xmit
ip_local_deliver
ip_push_pending_frames
ip_route_input
alloc_skb
ip_rcv
sk_buff
ip_output

drivers/
abstract devices and HID class drivers
drivers/input/
drivers/media/
sound/
console
kbd
fb_ops
mousedev
drm_driver

generic HW access
request_region
pci_driver
request_mem_region
pci_register_driver
pci_request_regions
usb_driver
usb_submit_urb
usb_hcd_giveback_urb
ioremap
usb_hcd

interrupts core
request_irq
tasklet_struct
jiffies_64++
tasklet_action
do_timer
setup_irq
tick_periodic
timer_interrupt
do_softirq
do_IRQ
irq_desc
softirq_init
include/asm/

Page Allocator
/proc/slabinfo
mm/slob.c
mm/slub.c
mm/slab.c
kmem_cache
__free_pages
kmem_cache_init
__free_one_page
kmem_cache_alloc
__get_free_pages
page
__alloc_pages
vm_stat
totalram_pages
try_to_free_pages

block devices
block/
gendisk
block_device_operations
request_queue
init_scsi
scsi_device
scsi_driver
sd_fops
usb_storage_driver
usb_stor_host_template

network interface
linux/netdevice.h
netif_receive_skb
dev_queue_xmit
register_netdev
net_device
dev_ioctl
alloc_netdev_mq
ieee80211_alloc_hw
ether_setup
ieee80211_rx
netif_carrier_on
ieee80211_xmit
drivers/net/

HI peripherals device drivers
uvc_driver
ac97_driver
vga_con
atkbd_drv
i8042_driver
psmouse
drivers/media/video/

device access and bus drivers
ehci_irq
writew
readw
ehci_urb_enqueue
outw
inw
usb_hcd_irq
pci_read
pci_write

CPU specific
arch/x86/
setup_arch
x86_init
trap_init
early_trap_init
start_thread
native_init_IRQ
switch_to
/proc/interrupts
set_intr_gate
system_call
show_regs
interrupt
pt_regs
atomic_t
cli
sti

physical memory operations
arch/x86/mm/
mem_init
get_page_from_freelist
zonelist
zone
free_area
free_list
out_of_memory
die
num_physpages
do_page_fault

disk controller drivers
scsi_host_alloc
libata
Scsi_Host
ahci_pci_driver
aic94xx_init

network device drivers
usbnet_probe
ipw2100_pci_init_one
zd1201_probe
e1000_xmit_frame
e1000_intr

user peripherals
keyboard
camera
mouse
graphics card
audio

I/O
I/O mem
PCI controller
I/O ports
ACPI
USB controller

CPU
registers
APIC
interrupt controller

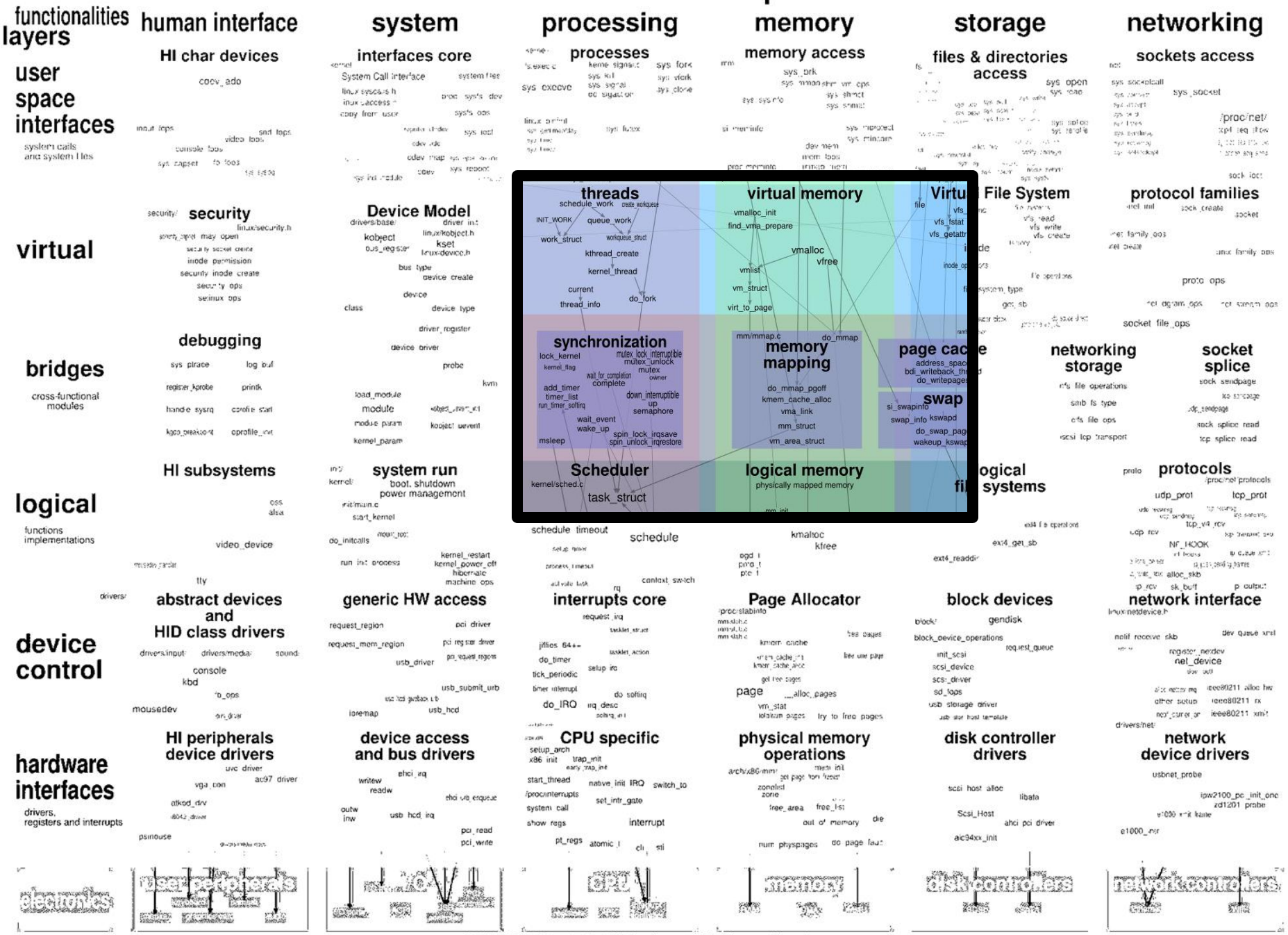
memory
RAM
DMA
MMU

disk controllers
SCSI
SATA

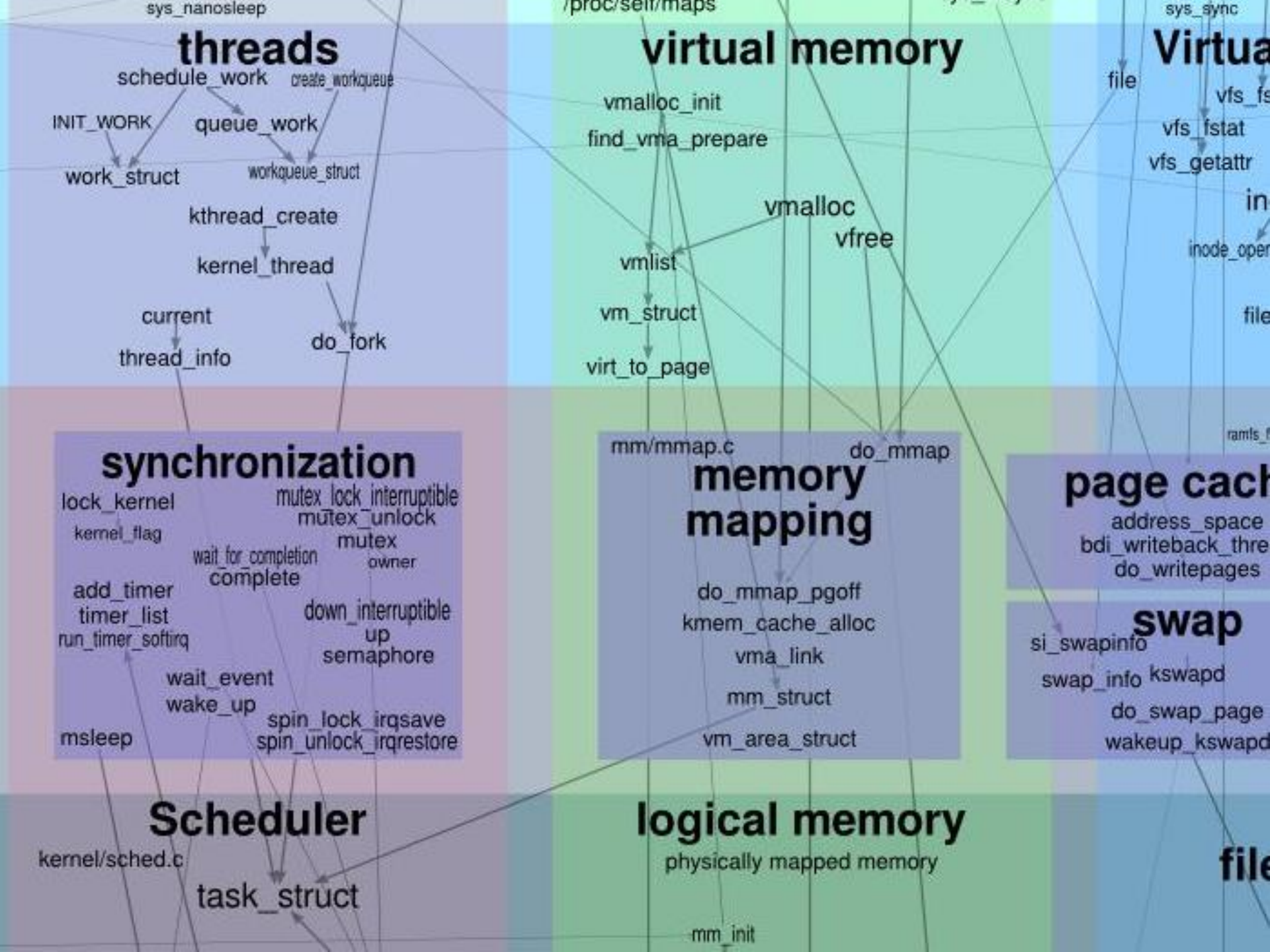
network controllers
Ethernet
WiFi

© 2007, 2010 Constantine Shulyupin www.MakeLinux.net/kernel_map

Linux kernel map

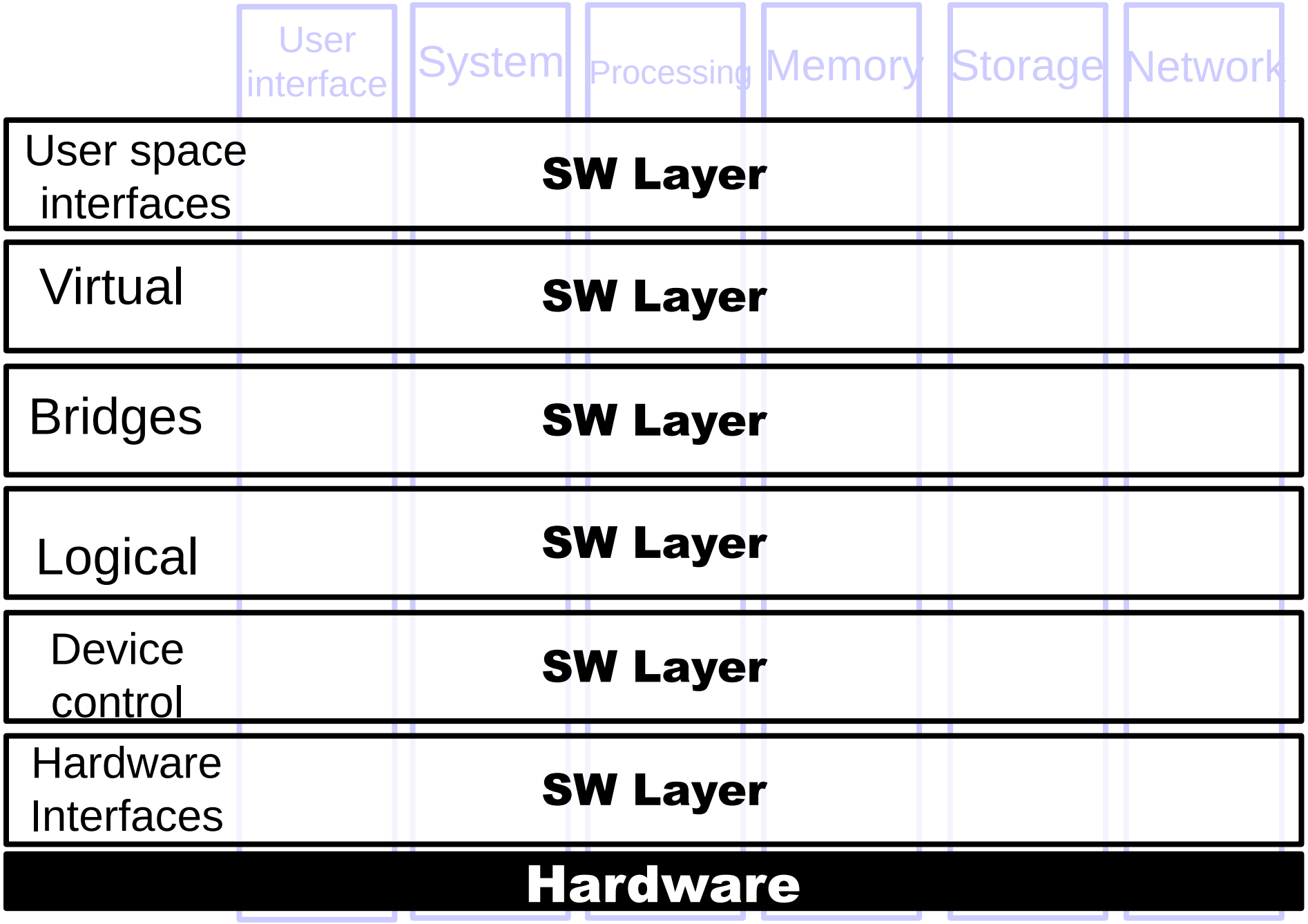


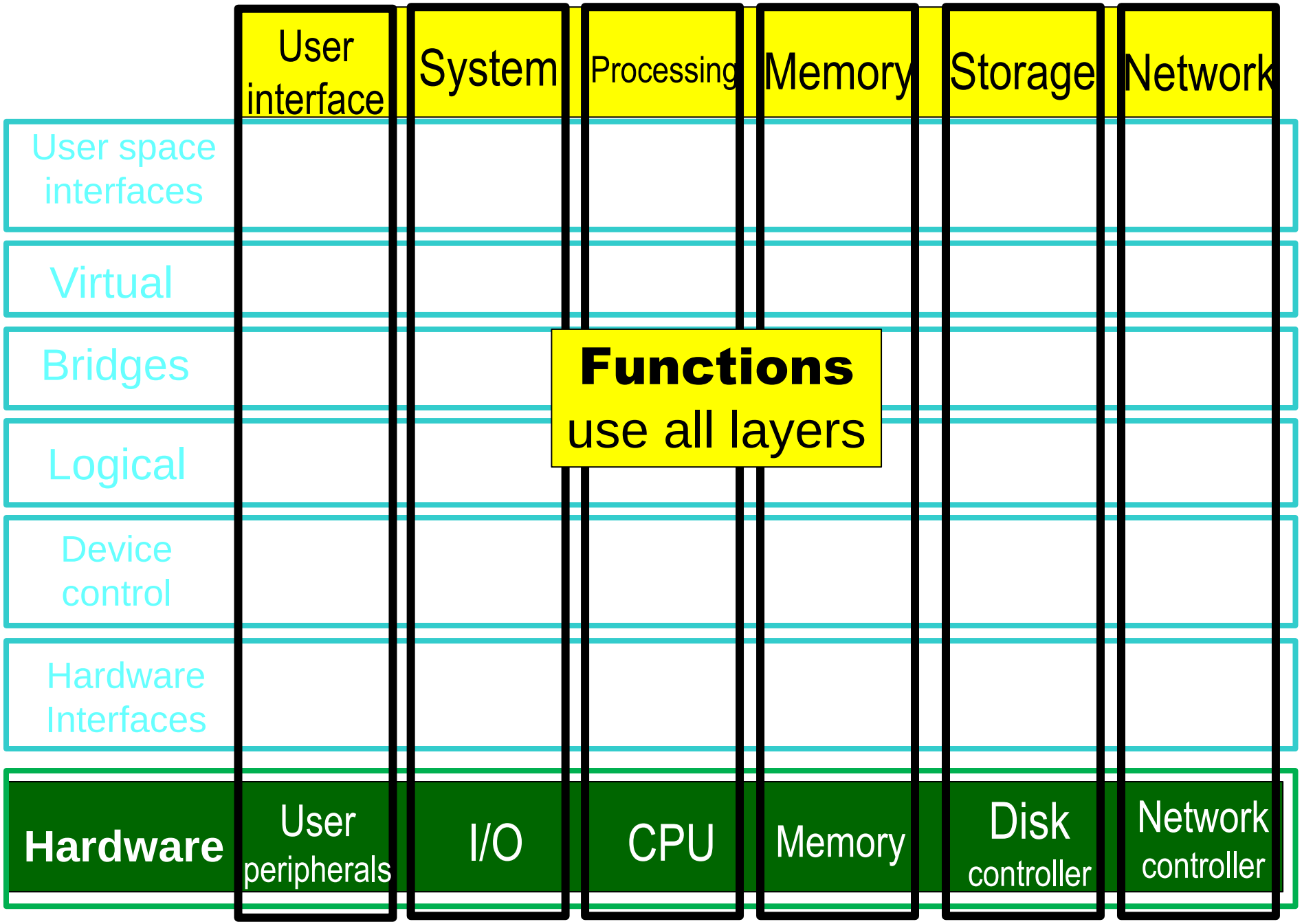
© 2007, 2010 Constantine Shulyupin www.MakeLinux.net/kernel_map


sys_nanosleep
threads
schedule_work
create_workqueue
INIT_WORK
queue_work
work_struct
workqueue_struct
kthread_create
kernel_thread
current
do_fork
thread_info
/proc/self/maps
virtual memory
vmalloc_init
find_vma_prepare
vmalloc
vfree
vmlist
vm_struct
virt_to_page
sys_sync
Virtua
file
vfs_fs
vfs_fstat
vfs_getattr
in
inode_oper
file
ramfs_f
synchronization
lock_kernel
kernel_flag
mutex_lock_interruptible
mutex_unlock
mutex
owner
wait_for_completion
complete
add_timer
timer_list
run_timer_softirq
down_interruptible
up
semaphore
wait_event
wake_up
spin_lock_irqsave
spin_unlock_irqrestore
msleep
mm/mmap.c
do_mmap
memory mapping
do_mmap_pgoff
kmem_cache_alloc
vma_link
mm_struct
vm_area_struct
page cach
address_space
bdi_writeback_thre
do_writepages
swap
si_swapinfo
swap_info
kswapd
do_swap_page
wakeup_kswapd
Scheduler
kernel/sched.c
task_struct
logical memory
physically mapped memory
mm_init
file

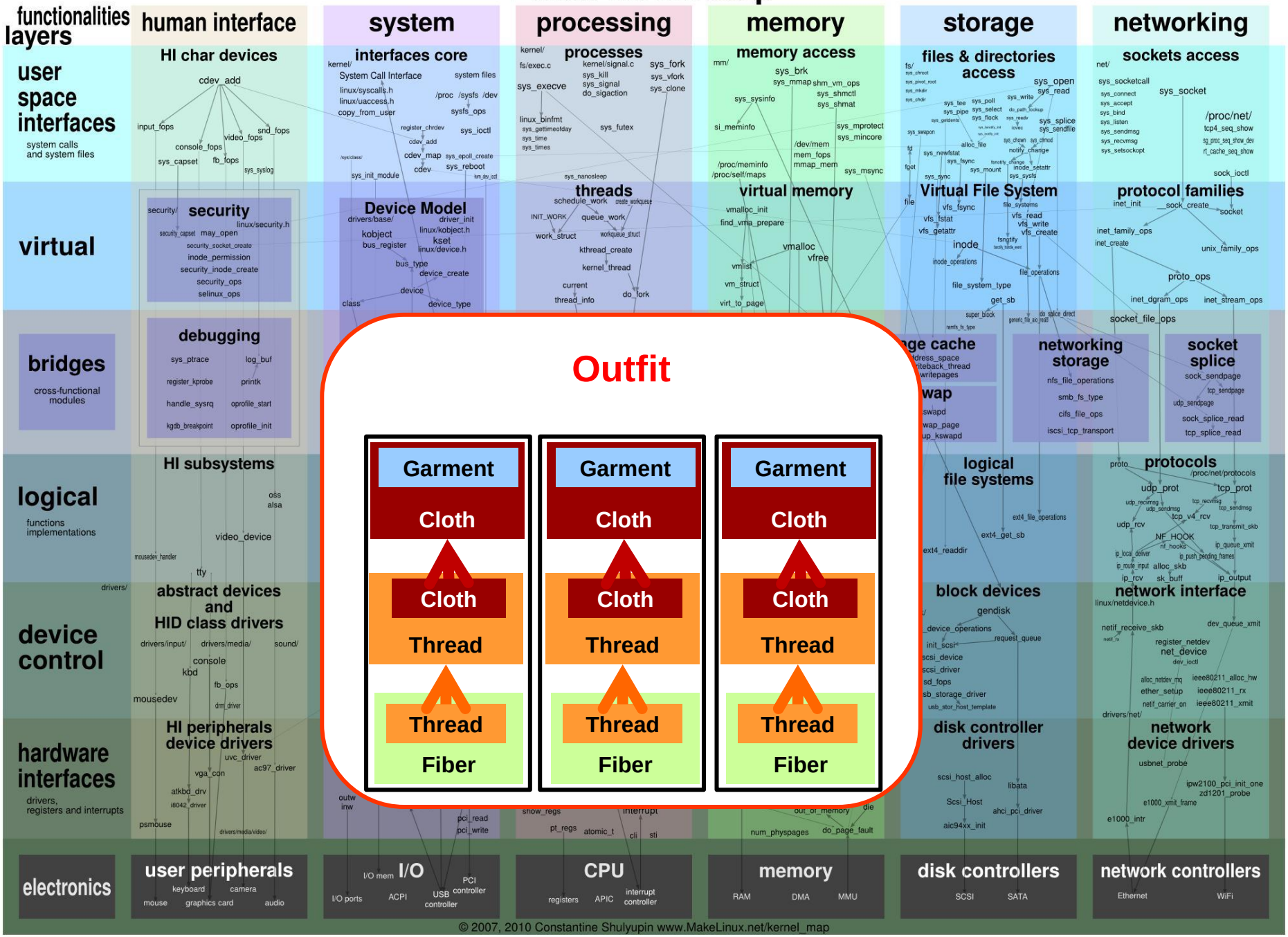
Linux kernel map

	User interface	System	Processing	Memory	Storage	Network
User space interfaces						
Virtual						
Bridges						
Logical						
Device control						
Hardware Interfaces						
Hardware						

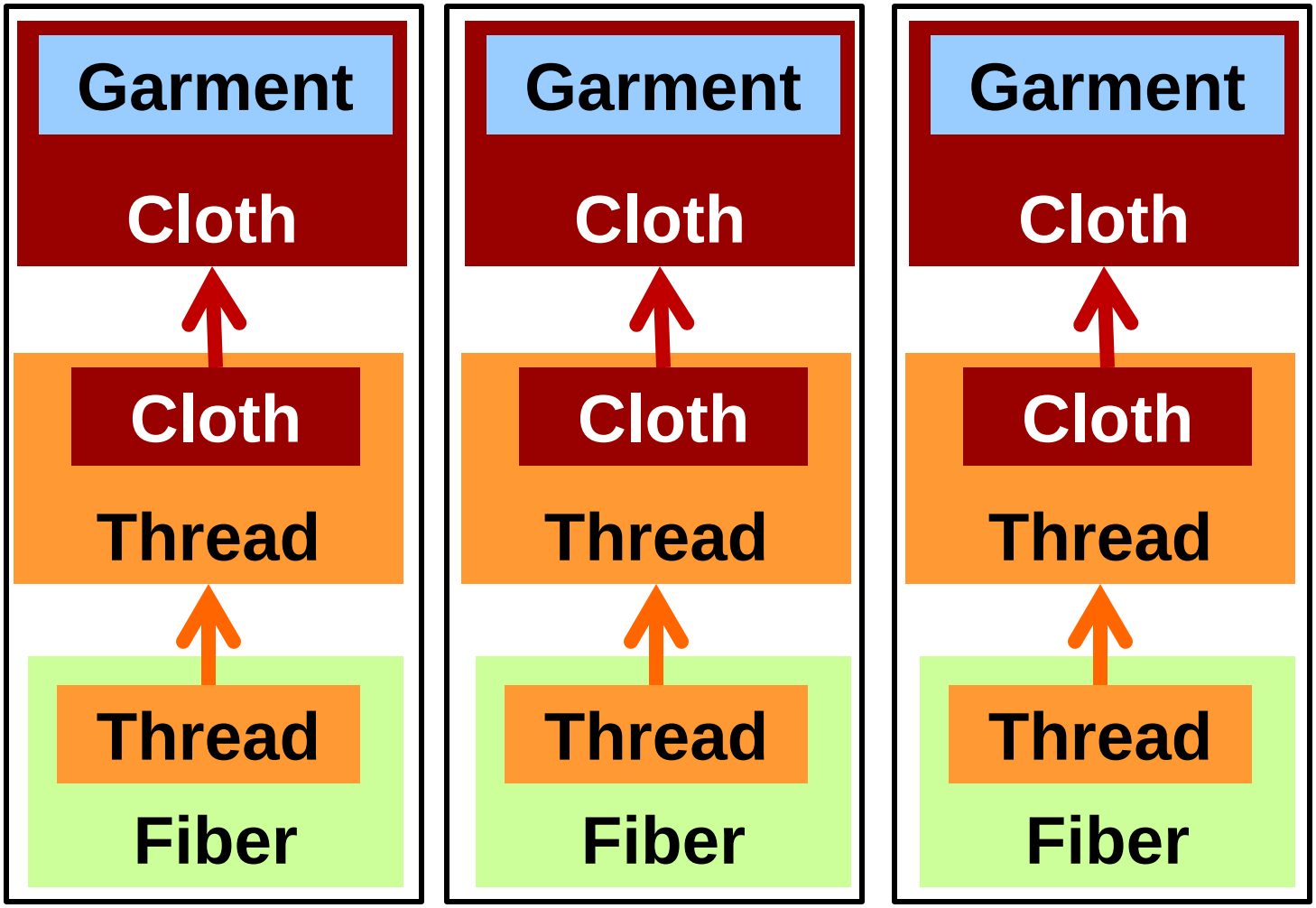

User interface
System
Processing
Memory
Storage
Network
User space interfaces
SW Layer
Virtual
SW Layer
Bridges
SW Layer
Logical
SW Layer
Device control
SW Layer
Hardware Interfaces
SW Layer
Hardware


User interface
System
Processing
Memory
Storage
Network
User space interfaces
Virtual
Bridges
Logical
Device control
Hardware Interfaces
Functions use all layers
Hardware
User peripherals
I/O
CPU
Memory
Disk controller
Network controller

Linux kernel map



Outfit


Garment
Cloth
Cloth
Thread
Thread
Fiber
Garment
Cloth
Cloth
Thread
Thread
Fiber
Garment
Cloth
Cloth
Thread
Thread
Fiber

Hycreek Layering Guide

This guide is meant as a "suggested" guide only. The amount and type of layers that will keep you warm in any given condition may vary, so you can feel free to adapt it as necessary.

Cool

Frigid

Layering

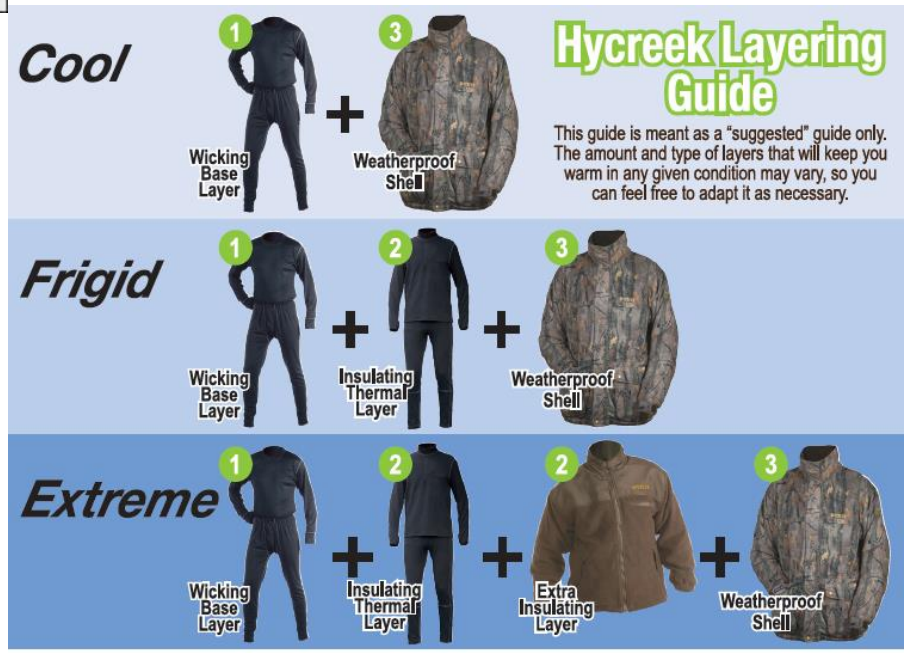
Tradeoffs

Extreme

functions

Extreme
1
Wicking Base Layer
2
Insulating Thermal Layer
2
Extra Insulating Layer
3
Weatherproof Shell

fragile



robust

cheap

costly

Layering & Modularity 1.0

Body

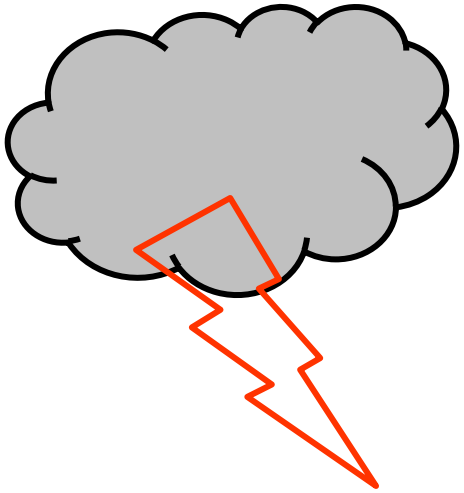
Outfit

Base

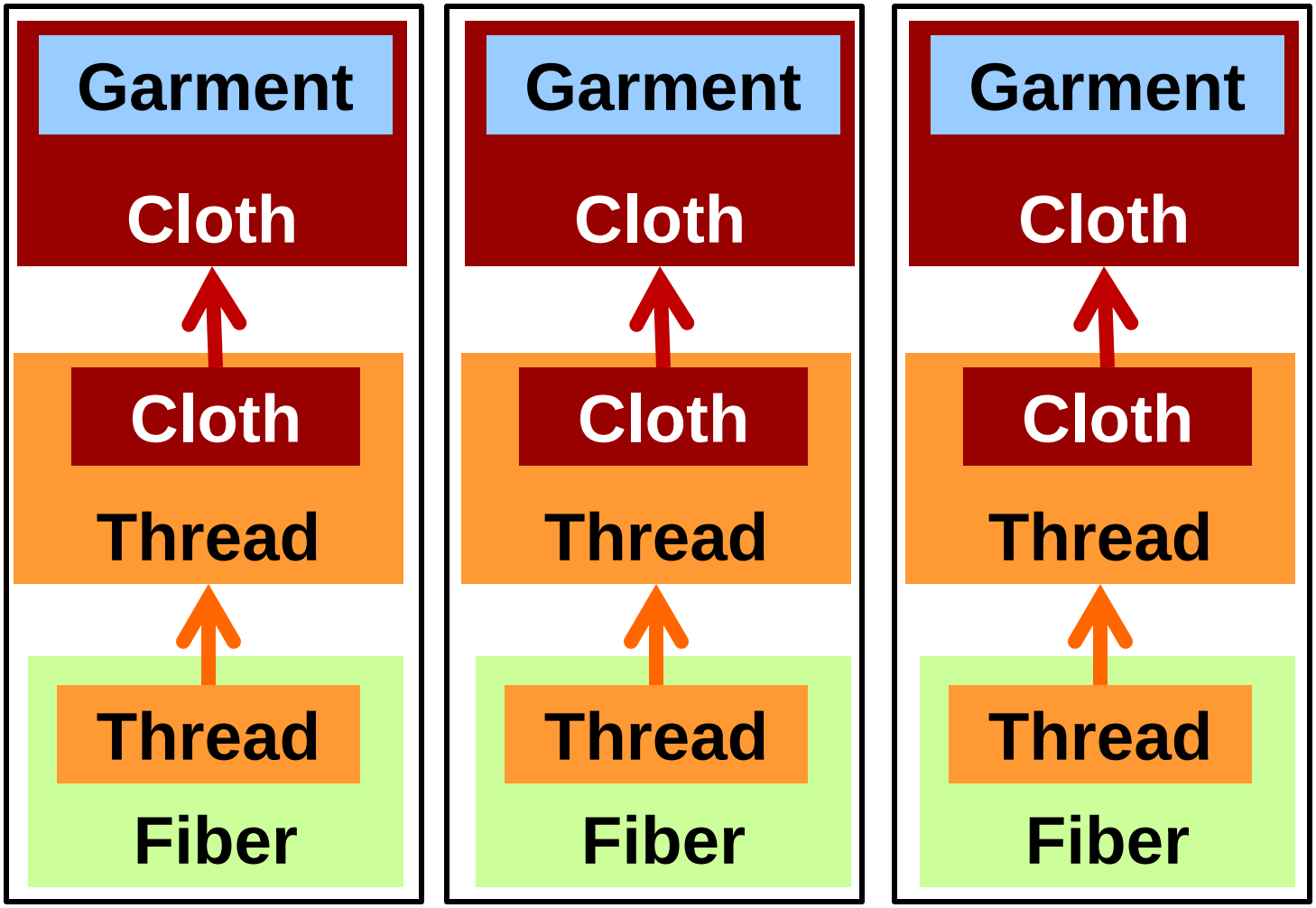
Insulation

Shell

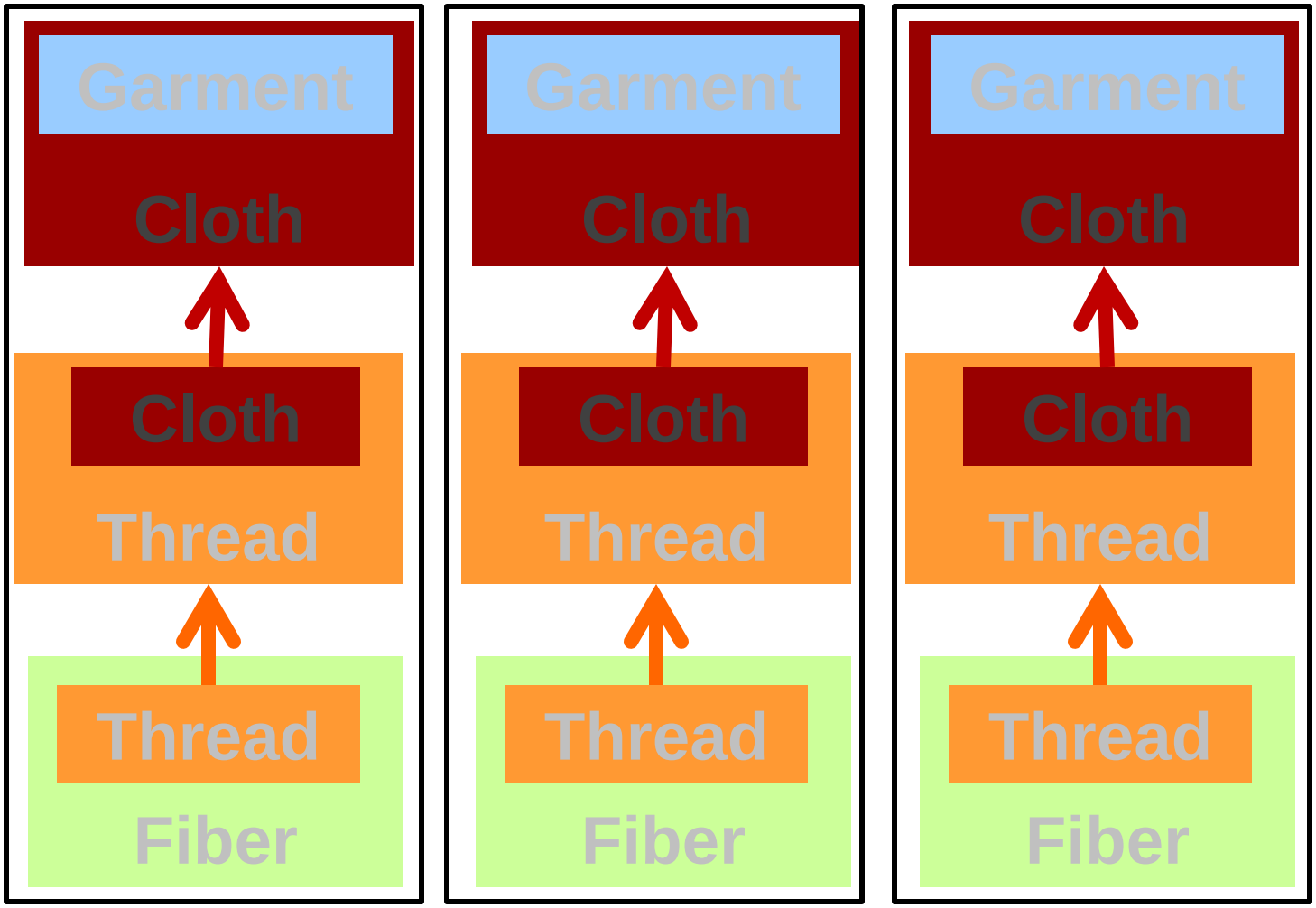
Environment



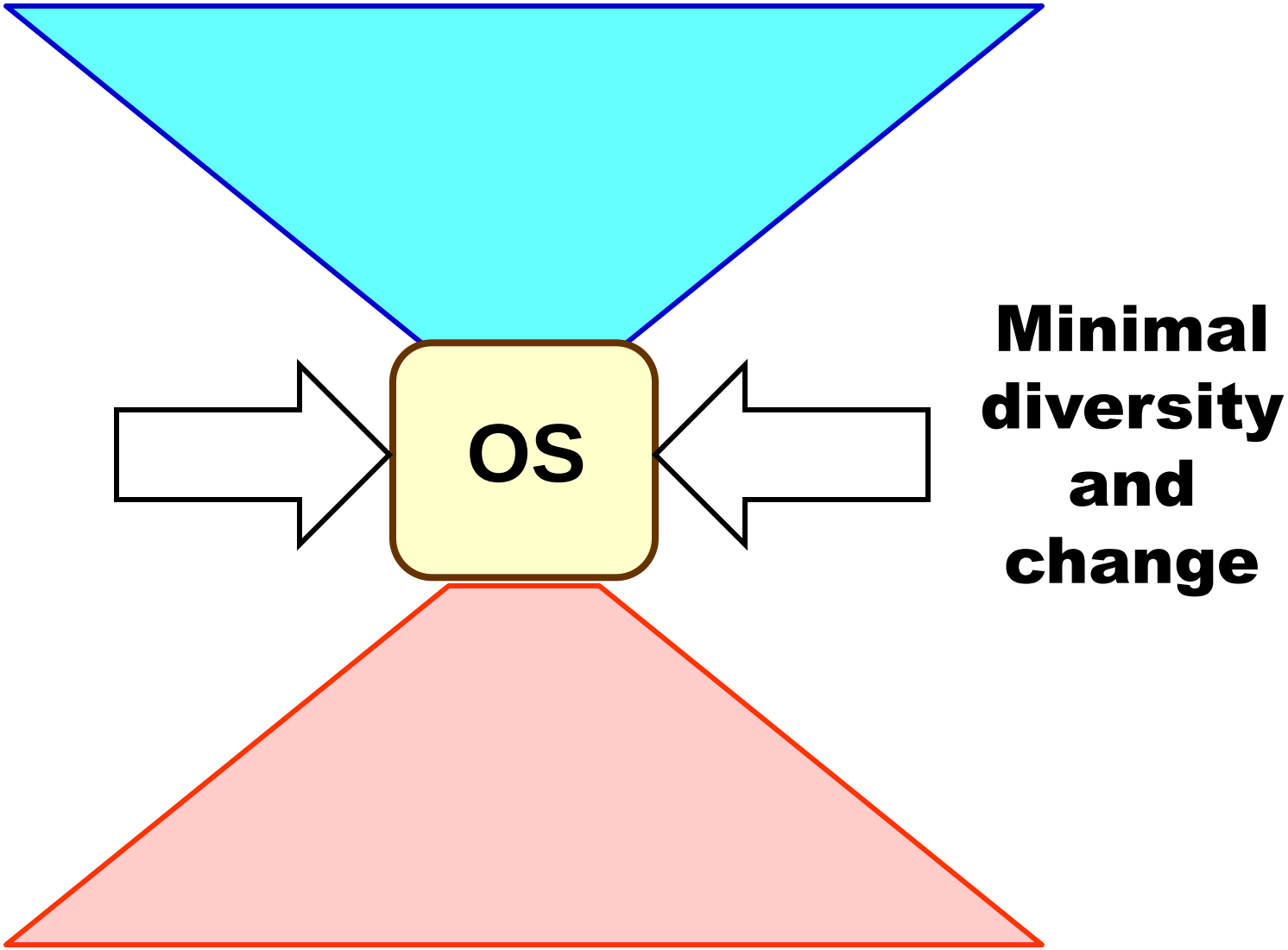
Outfit


Garment
Cloth
Cloth
Thread
Thread
Fiber
Garment
Cloth
Cloth
Thread
Thread
Fiber
Garment
Cloth
Cloth
Thread
Thread
Fiber

Architecture, Layering & Modularity 2.0



Layered architectures



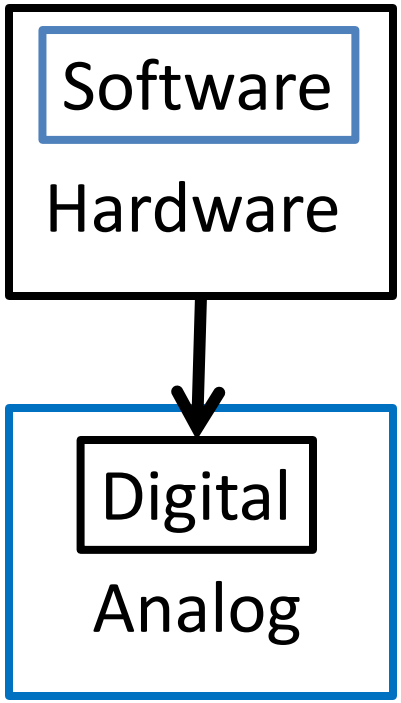
Minimal diversity and change

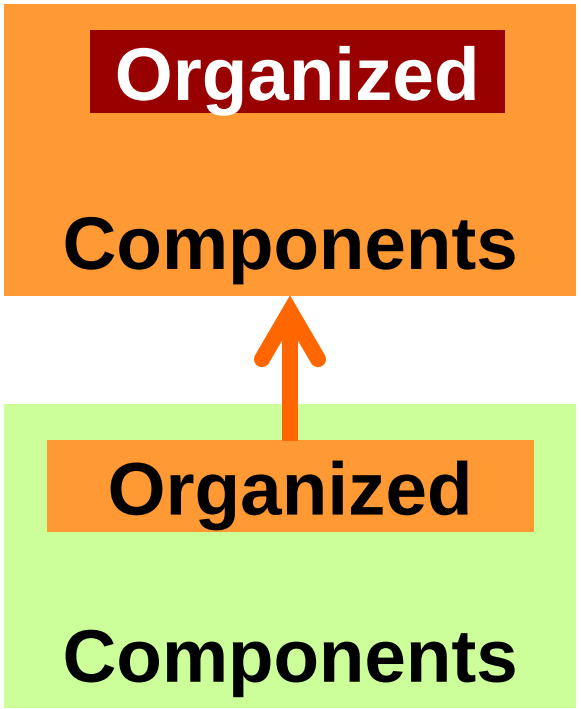
Apps

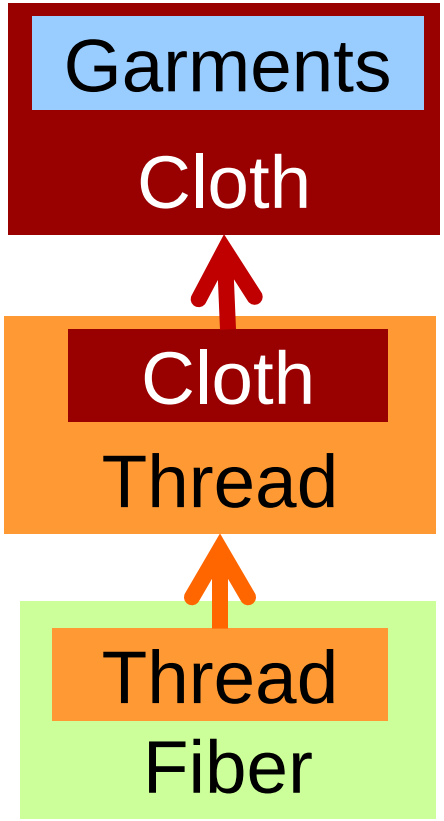
OS

HW

What is the "true functional ~~unit~~ Architecture?"







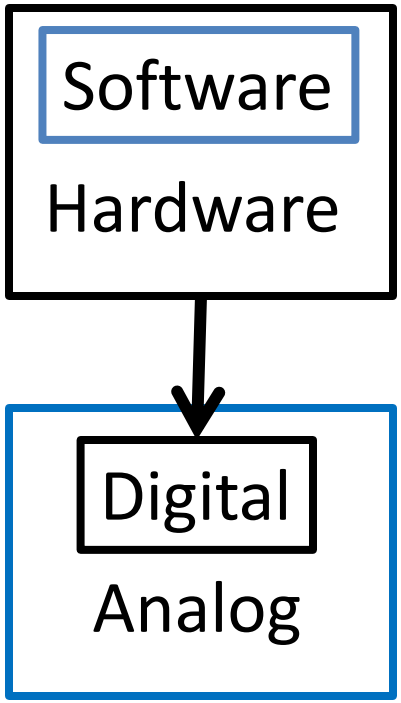
Virtual machines

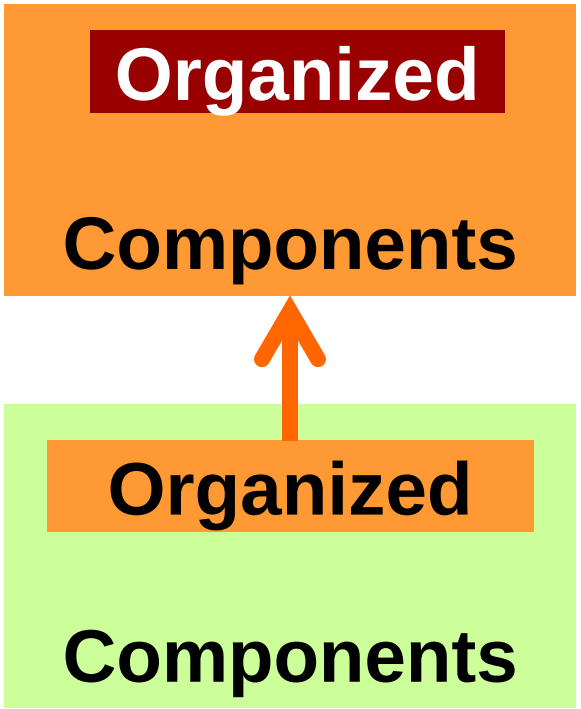
Software

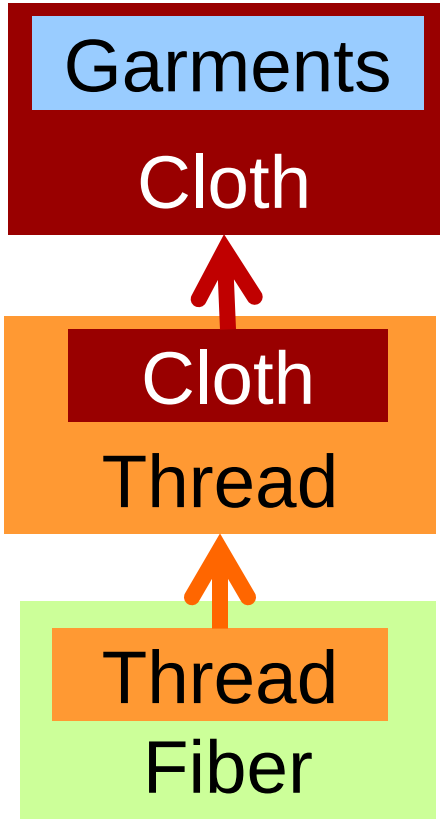
Organized

Garments

Implementations







Software

Organized

Garments

Virtual machines

Digital

Organized

Cloth

Implementations

Digital

Analog

Organized

Components

Cloth

Thread

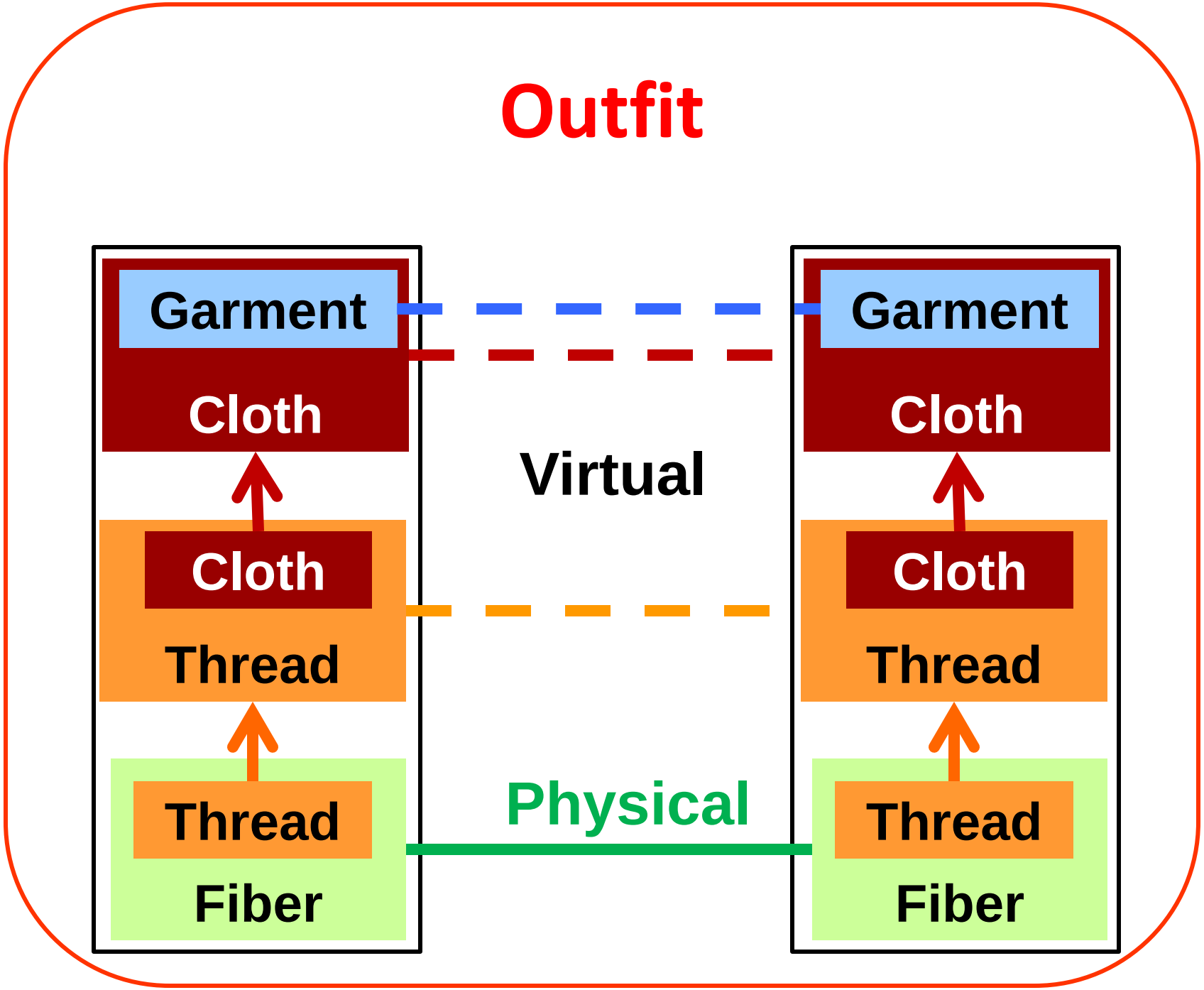
Thread

Fiber

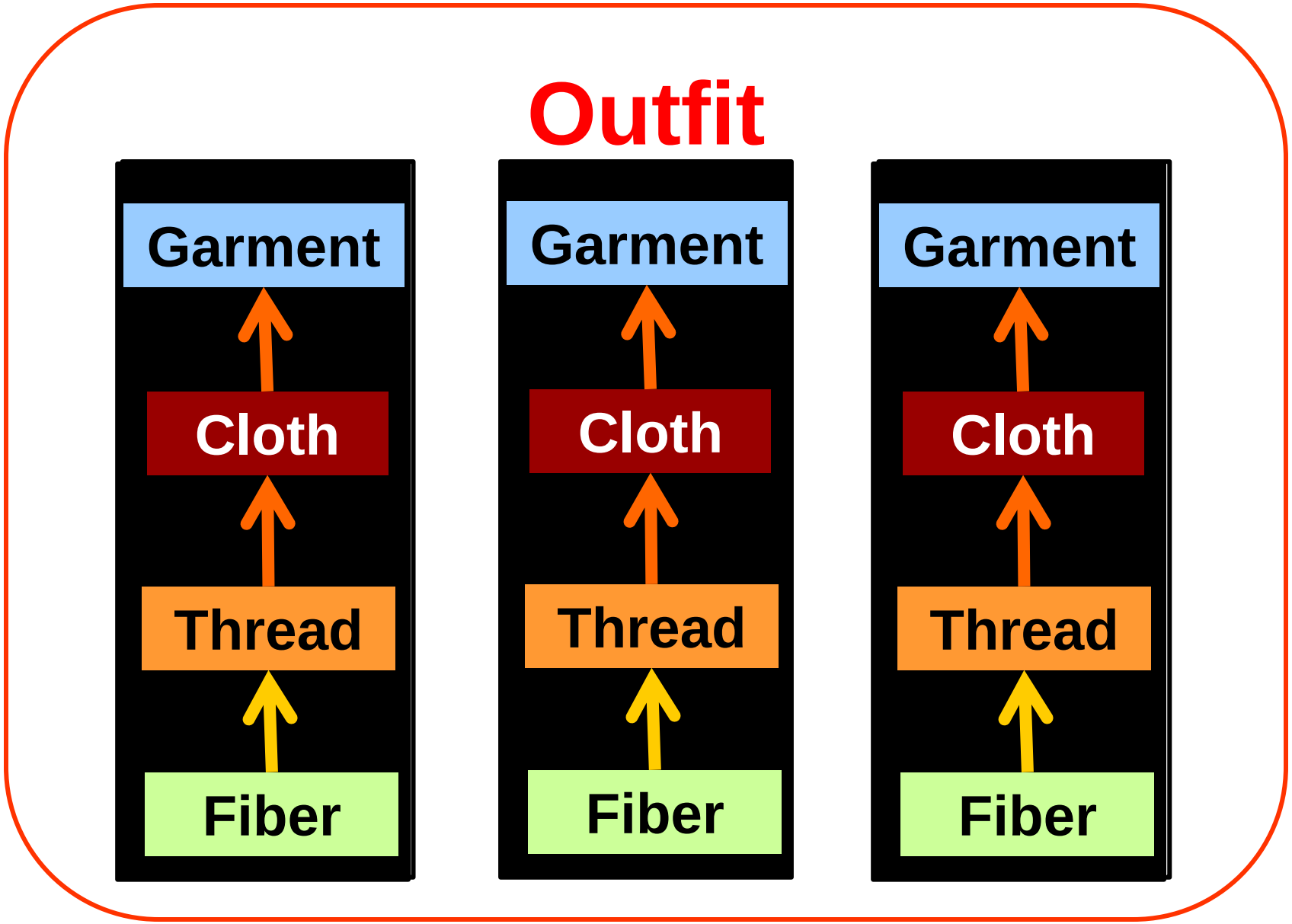
Virtual Outfit

Garment – Garment – Garment

Physical connectivity


Outfit
Garment
Cloth
Cloth
Thread
Thread
Fiber
Virtual
Physical
Garment
Cloth
Cloth
Thread
Thread
Fiber

Layering within garments (textiles)



Doyle, Csete, *Proc Nat Acad Sci USA*, JULY 25 2011

Architecture, constraints, and behavior

John C. Doyle[a,1] and Marie Csete[b,1]

[a]Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125; and [b]Department of Anesthesiology, University of California, San Diego, CA 92103

Edited by Donald W. Pfaff, The Rockefeller University, New York, NY, and approved June 10, 2011 (received for review March 3, 2011)

This paper aims to bridge progress in neuroscience involving sophisticated quantitative analysis of behavior, including the use of robust control, with other relevant conceptual and theoretical frameworks from systems engineering, systems biology, and mathematics. Familiar and accessible case studies are used to illustrate concepts of robustness, organization, and architecture (modularity and protocols) that are central to understanding complex networks. These essential organizational features are hidden during normal function of a system but are fundamental for understanding the nature, design, and function of complex biologic and technologic systems.

evolved for sensorimotor control and retain much of that evolved architecture, then the apparent distinctions between perceptual, cognitive, and motor processes may be another form of illusion (9), reinforcing the claim that robust control and adaptive feedback (7, 11) rather than more conventional serial signal processing might be more useful in interpreting neurophysiology data (9). This view also seems broadly consistent with the arguments from grounded cognition that modal simulations, bodily states, and situated action underlie not only motor control but cognition in general (12), including language (13). Furthermore, the myriad constraints involved in the evolution of circuit

For more info
Most accessible
No math
References

Implementation
Control of function

Slow

Apps

HW

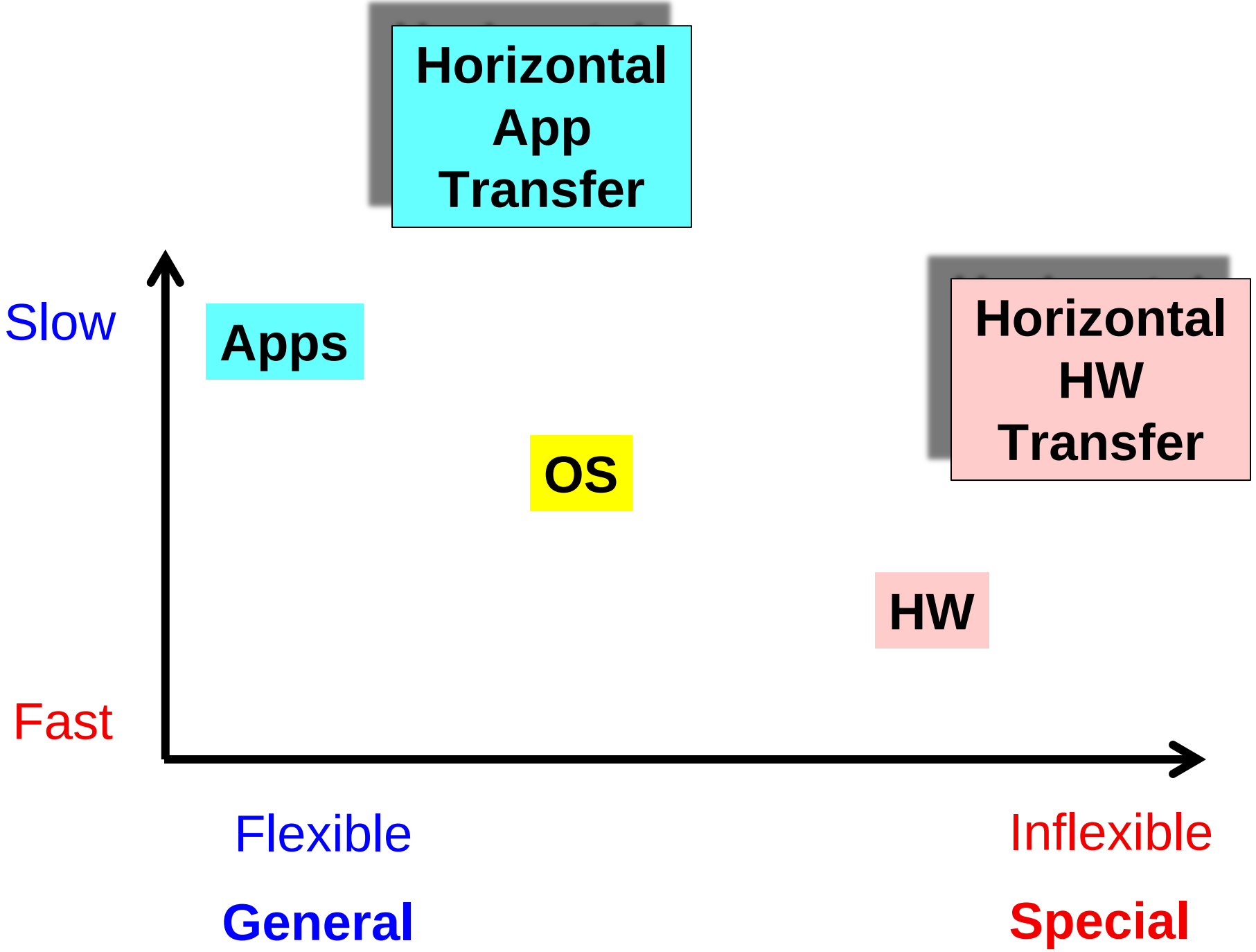
Fast

Flexible

Inflexible

General

Special


Horizontal App Transfer
Horizontal HW Transfer
Slow
Fast
Apps
OS
HW
Flexible
Inflexible
General
Special

Layered Architecture

Slow
Cheap

Apps

OS

HW

Digital

Lumped

Distrib.

Fast
Costly

Any layer needs
all lower layers.

Flexible

General

Inflexible

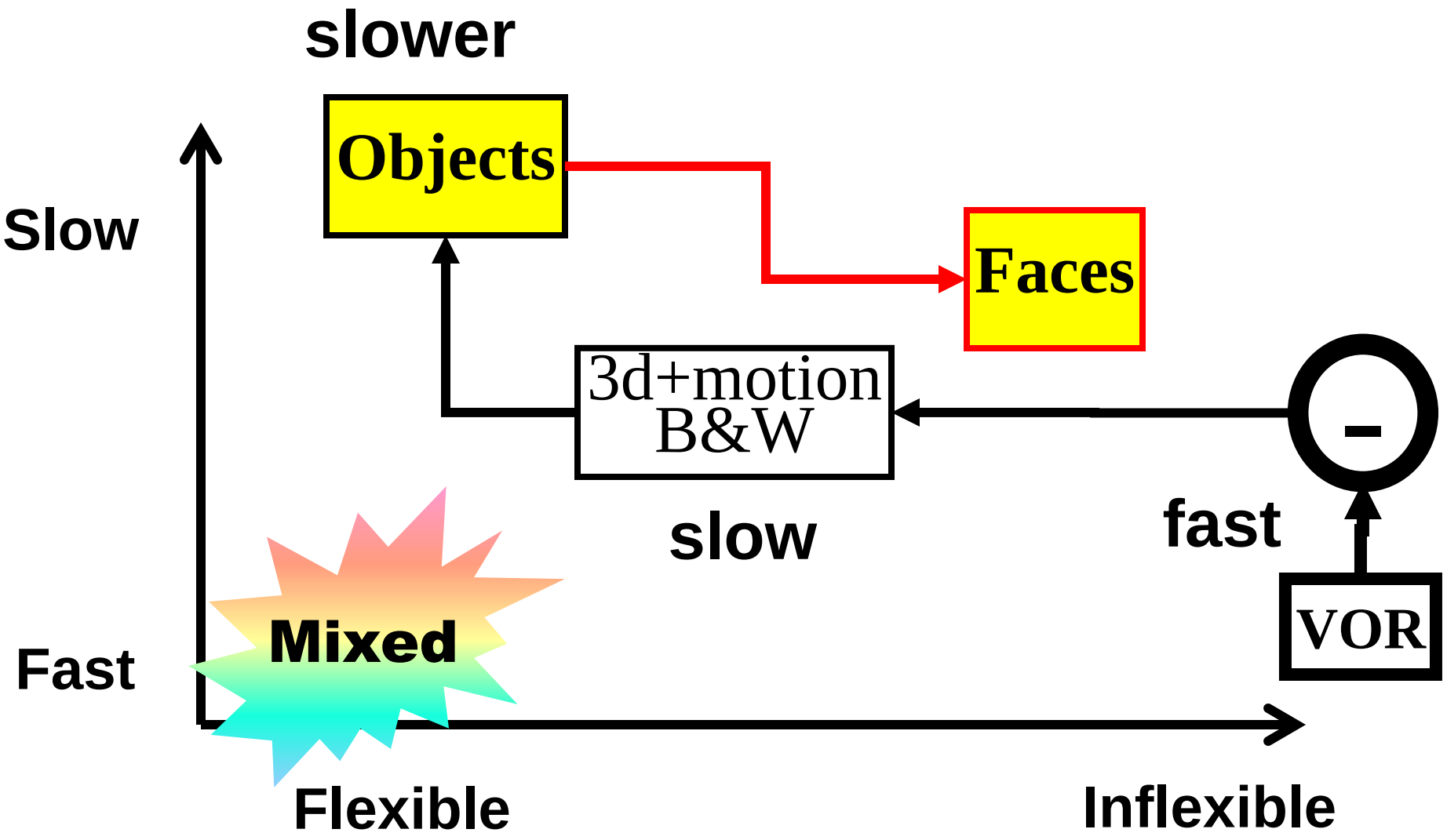
Special

You can control function
in hardware alone.

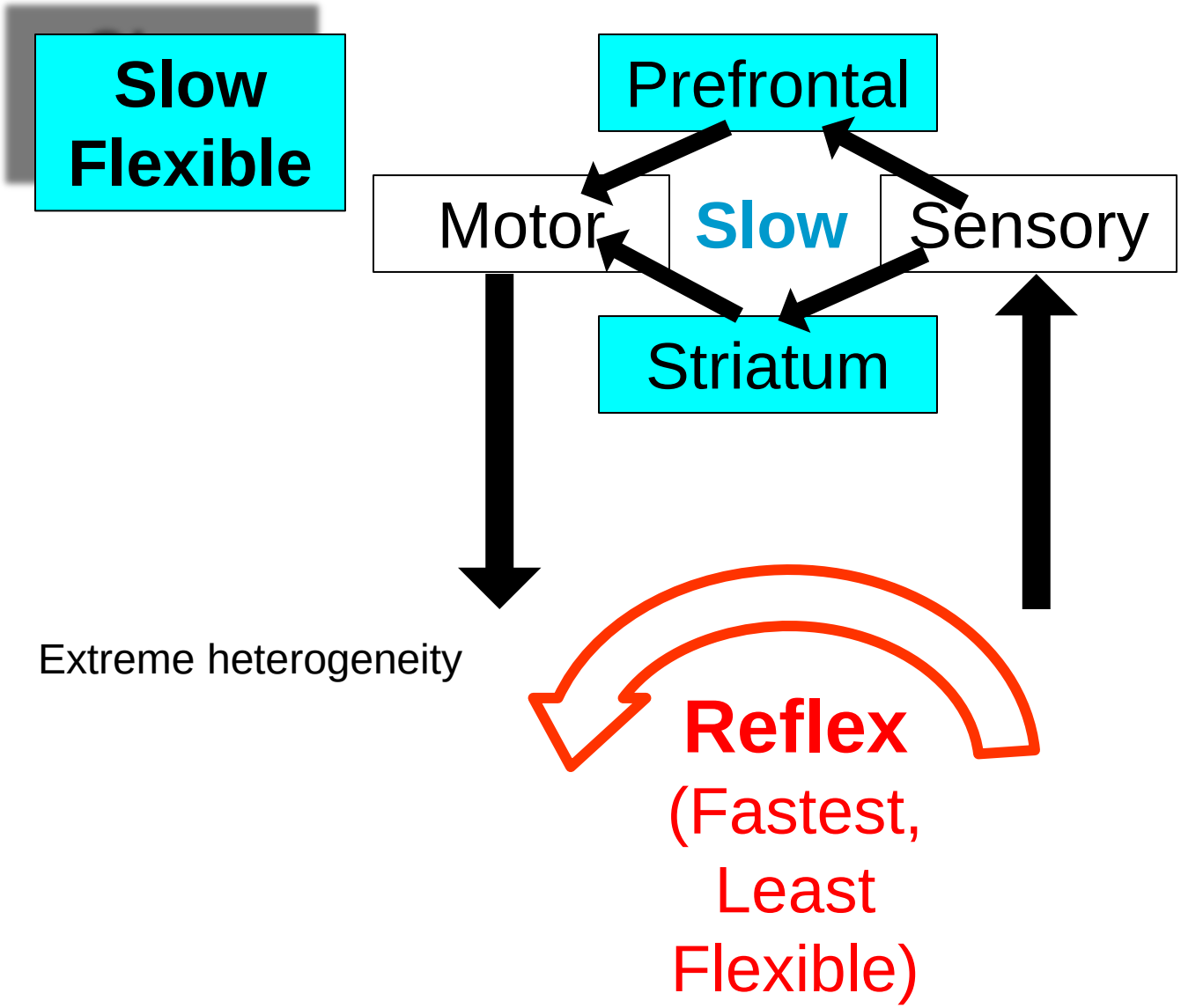
Hardware
Digital
Lumped
Distributed

Digital		
Lumped	**Lumped**	
Distrib.	Distrib.	**Distrib.**

Not sure how to draw this…

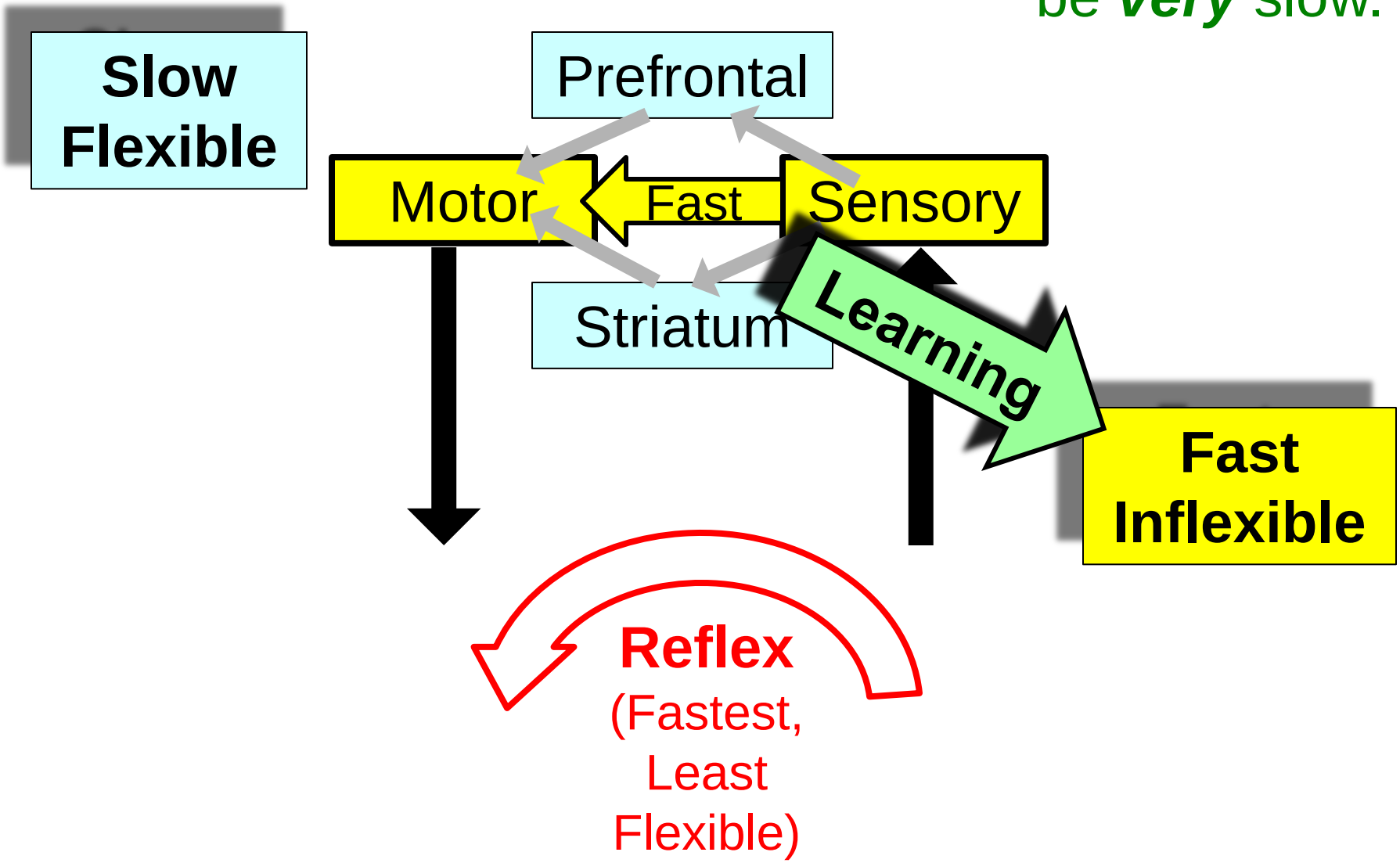


Learning



Ashby & Crossley

Learning can be ***very*** slow.

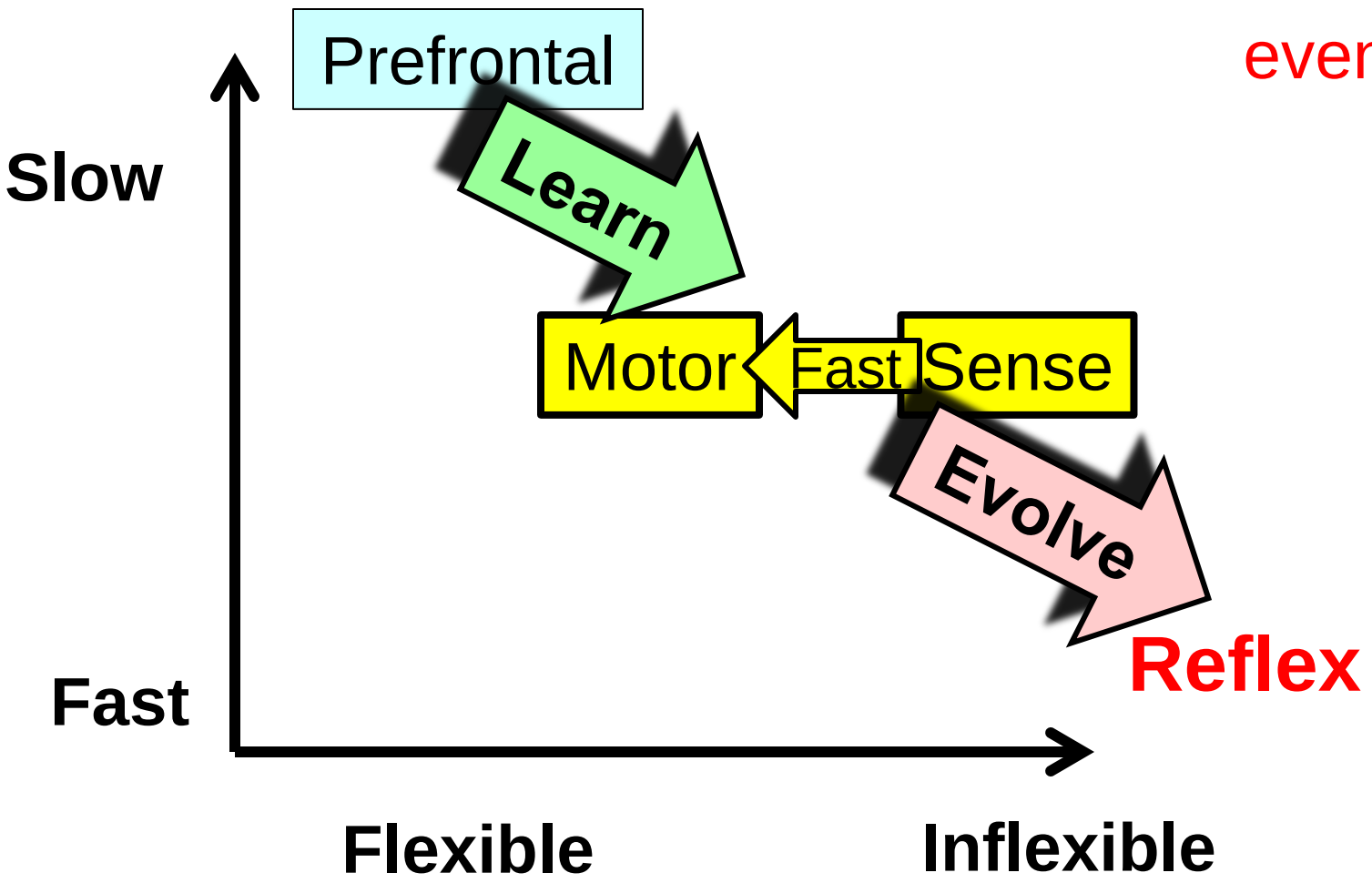


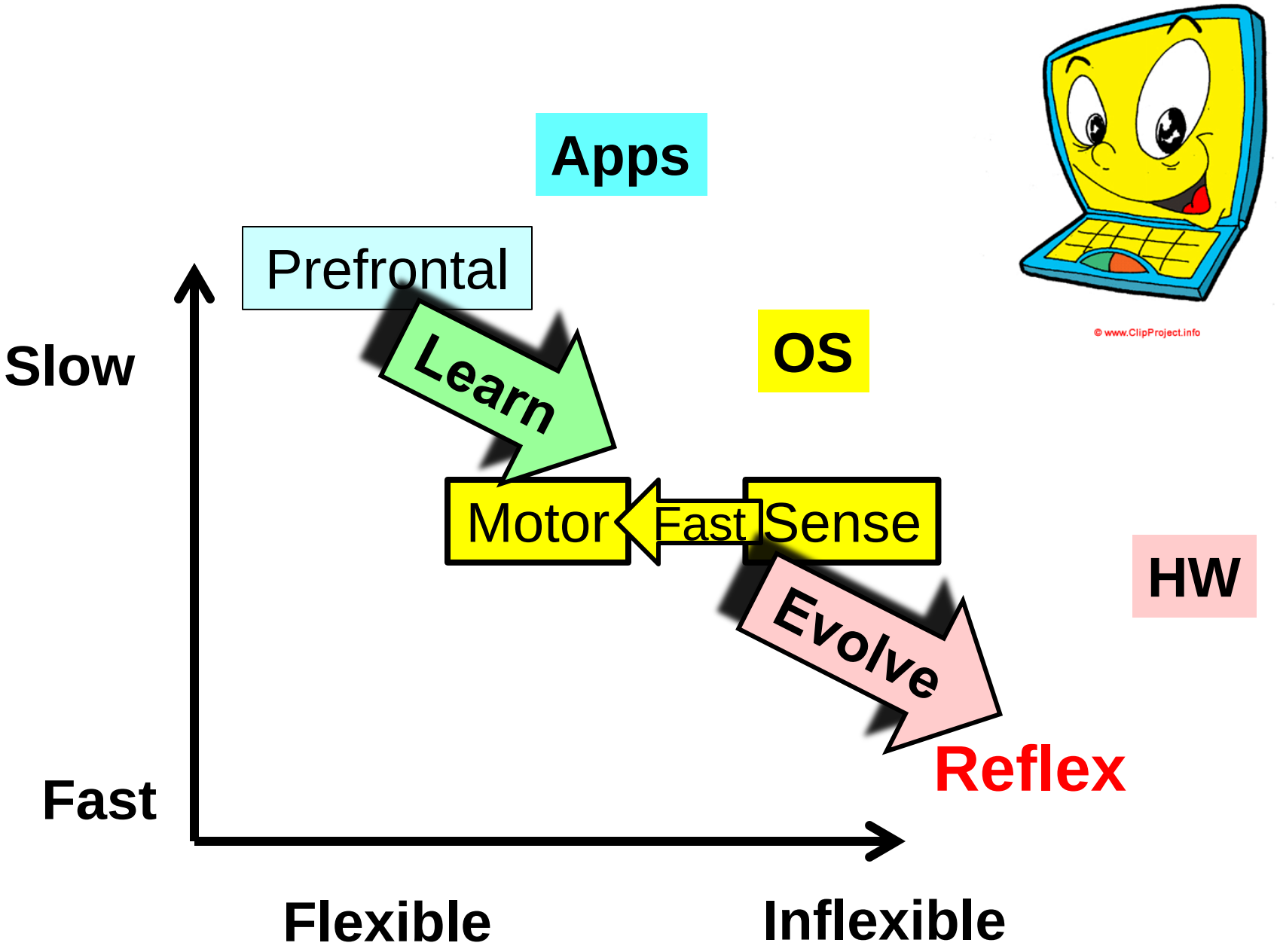
Ashby & Crossley

Universal tradeoffs?

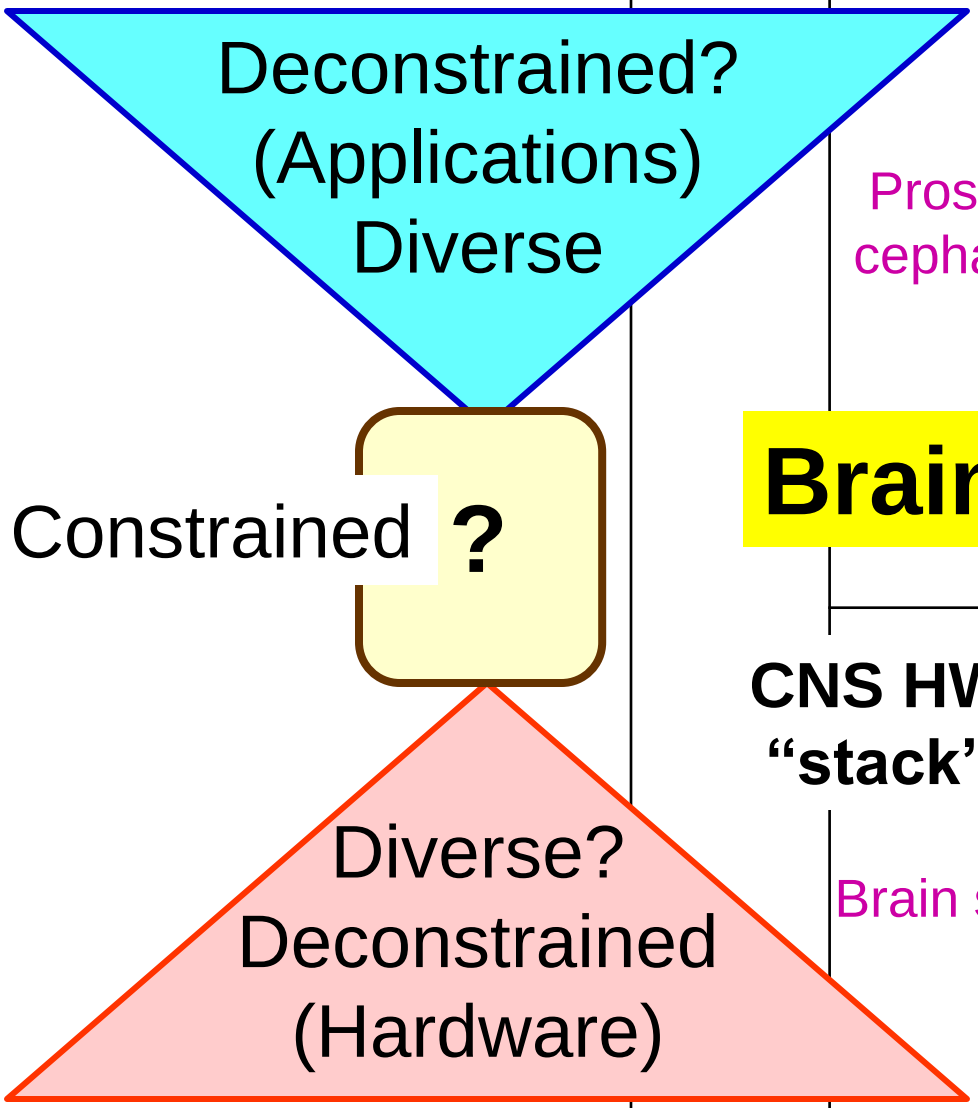
Learning can be ***very*** slow.

Evolution is even slower.



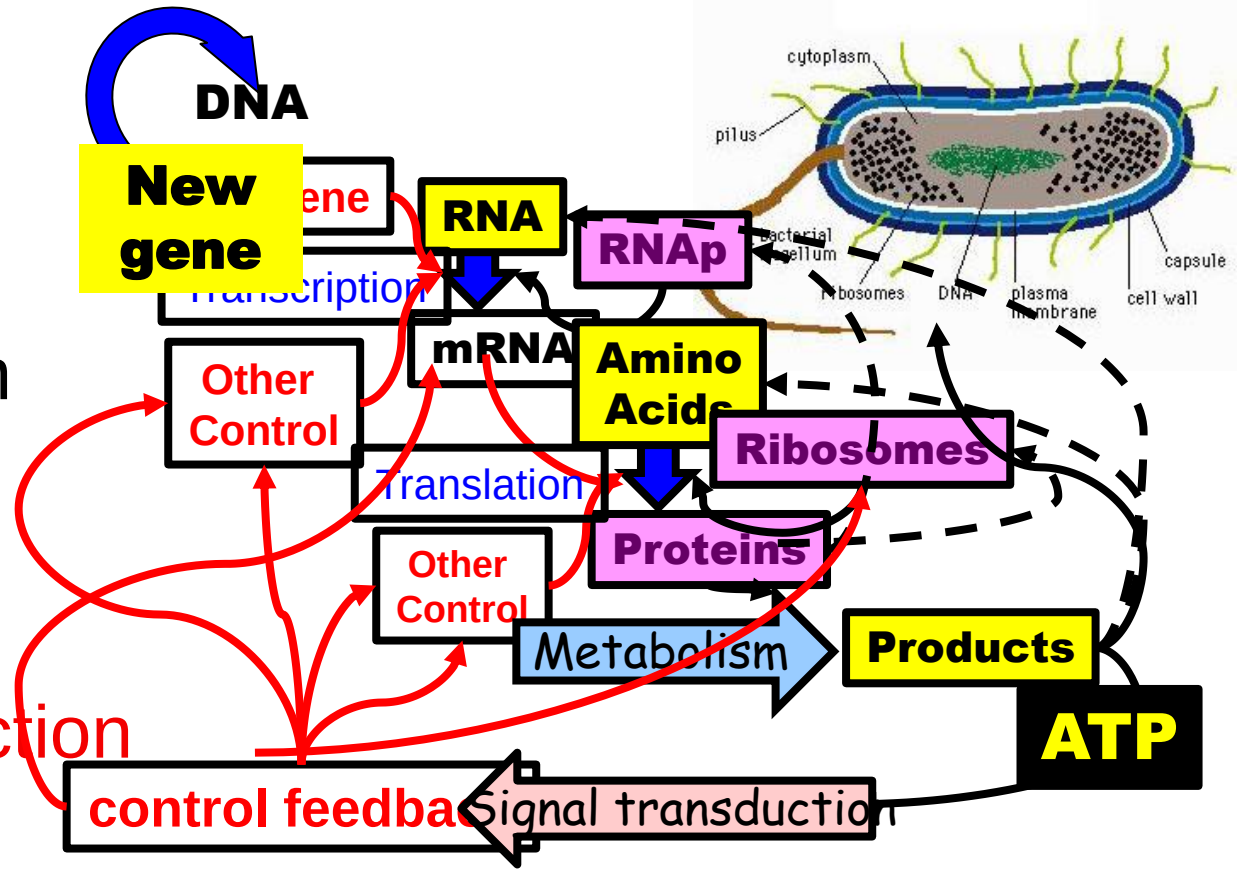

Apps
Prefrontal
Slow
Learn
OS
Motor
Fast
Sense
HW
Evolve
Reflex
Fast
Flexible
Inflexible
© www.ClipProject.info

Layered architectures??

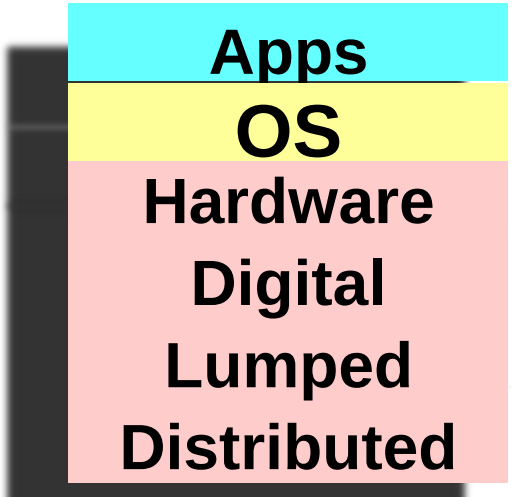


Brain	Prosen-cephalon	Telen-cephalon	Rhinencephalon, Amygdala, Hippocampus, Neocortex, Basal ganglia, Lateral ventricles
		Dien-cephalon	Epithalamus, Thalamus, Hypothalamus, Subthalamus, Pituitary gland, Pineal gland, Third ventricle
CNS HW "stack"	Brain stem	Mesen-cephalon	Tectum, Cerebral peduncle, Pretectum, Mesencephalic duct
		Rhomb-encephalon	Meten-cephalon: Pons, Cerebellum
			Myelen-cephalon: Medulla oblongata
Spinal cord			

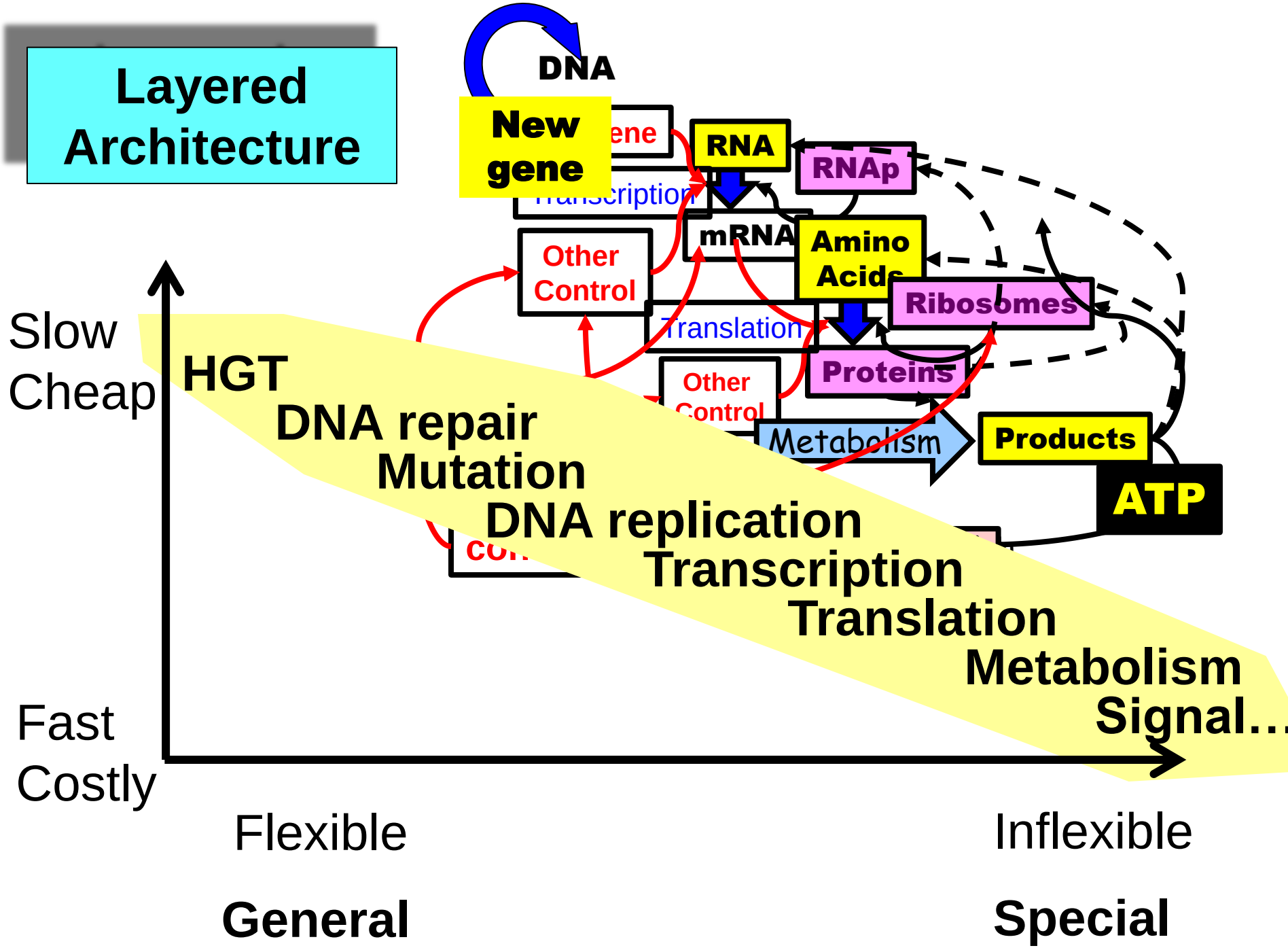
0. **HGT**
1. DNA repair
2. Mutation
3. DNA replication
4. Transcription
5. Translation
6. Metabolism
7. Signal transduction
8. …

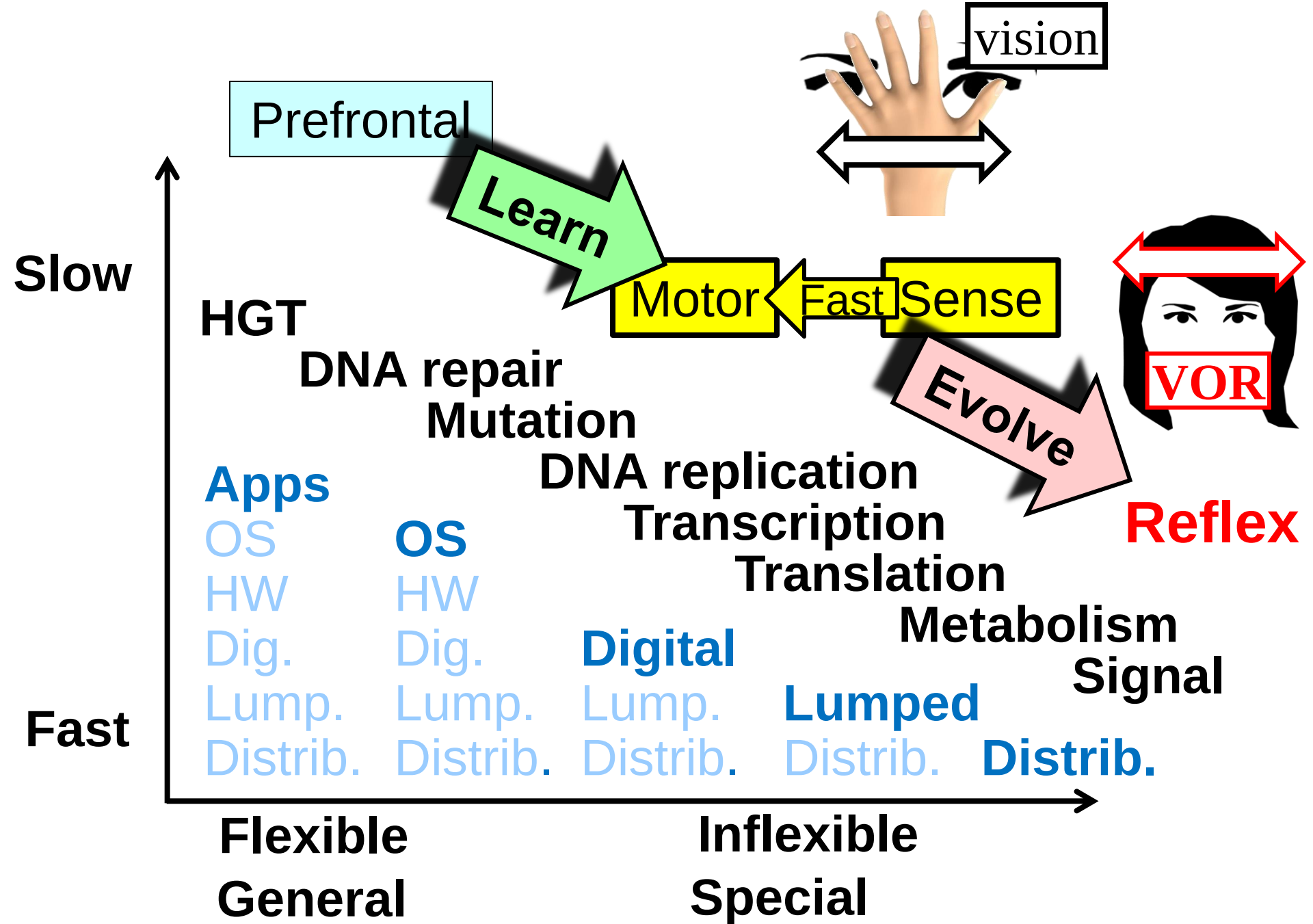


Any layer needs all lower layers.

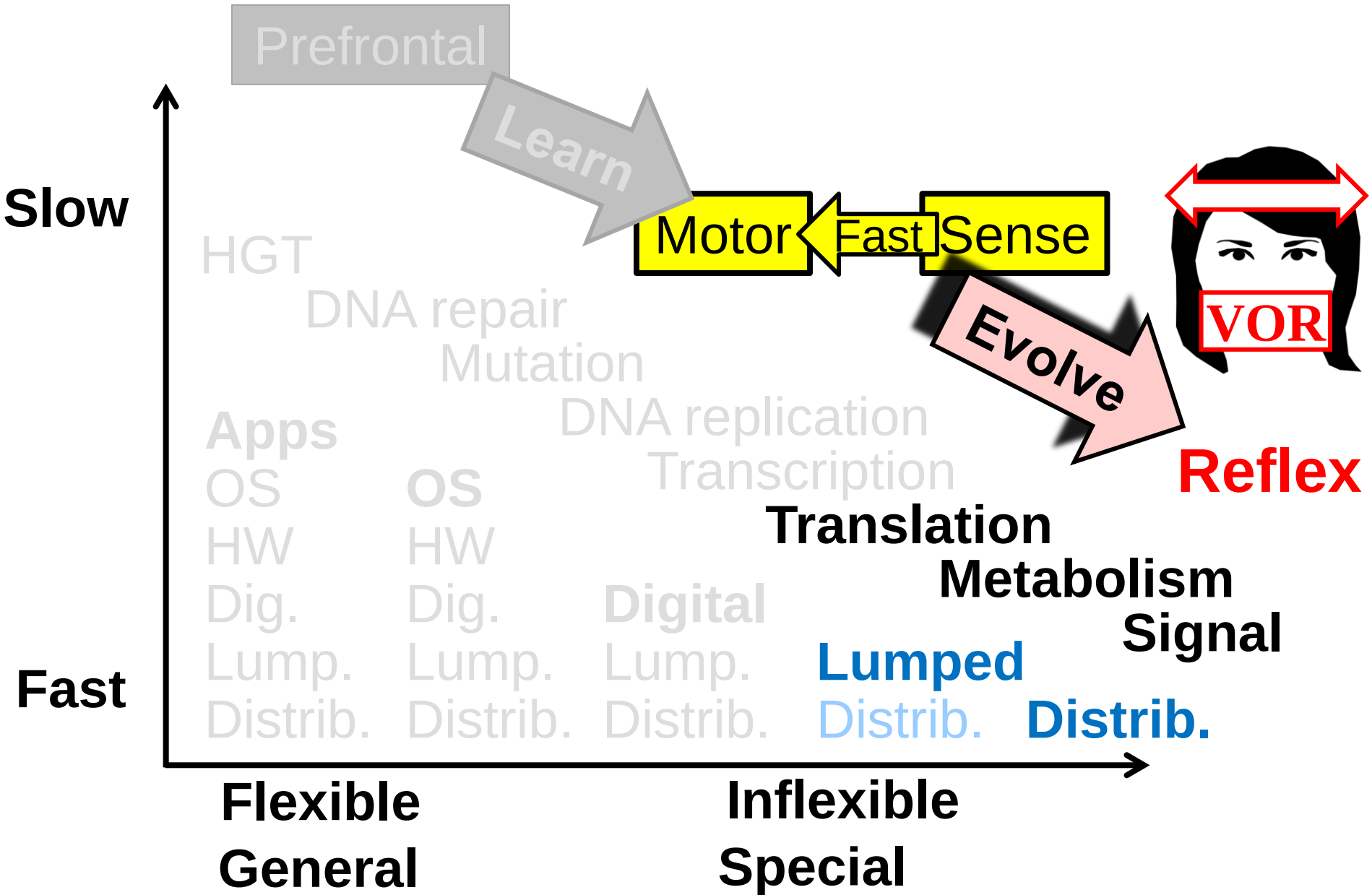


Layered Architecture

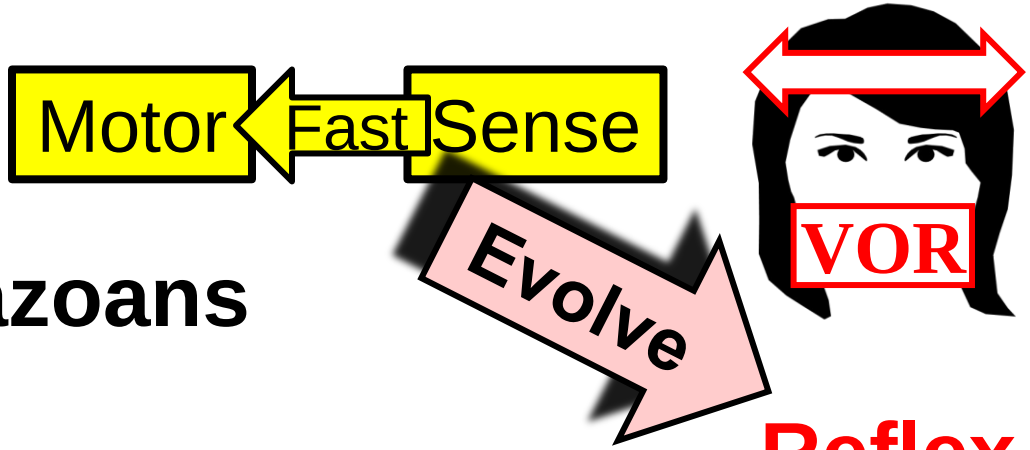



vision
Prefrontal
Learn
Motor
Fast
Sense
Evolve
VOR
Slow
Fast
HGT
DNA repair
Mutation
DNA replication
Transcription
Translation
Metabolism
Signal
Reflex
Apps
OS
HW
Dig.
Lump.
Distrib.
OS
HW
Dig.
Lump.
Distrib.
Digital
Lump.
Distrib.
Lumped
Distrib.
Distrib.
Flexible
Inflexible
General
Special

Many systems/organisms only use lower layers



Many systems/organisms only use lower layers



Most metazoans

Reflex

Red blood cells

Metabolism

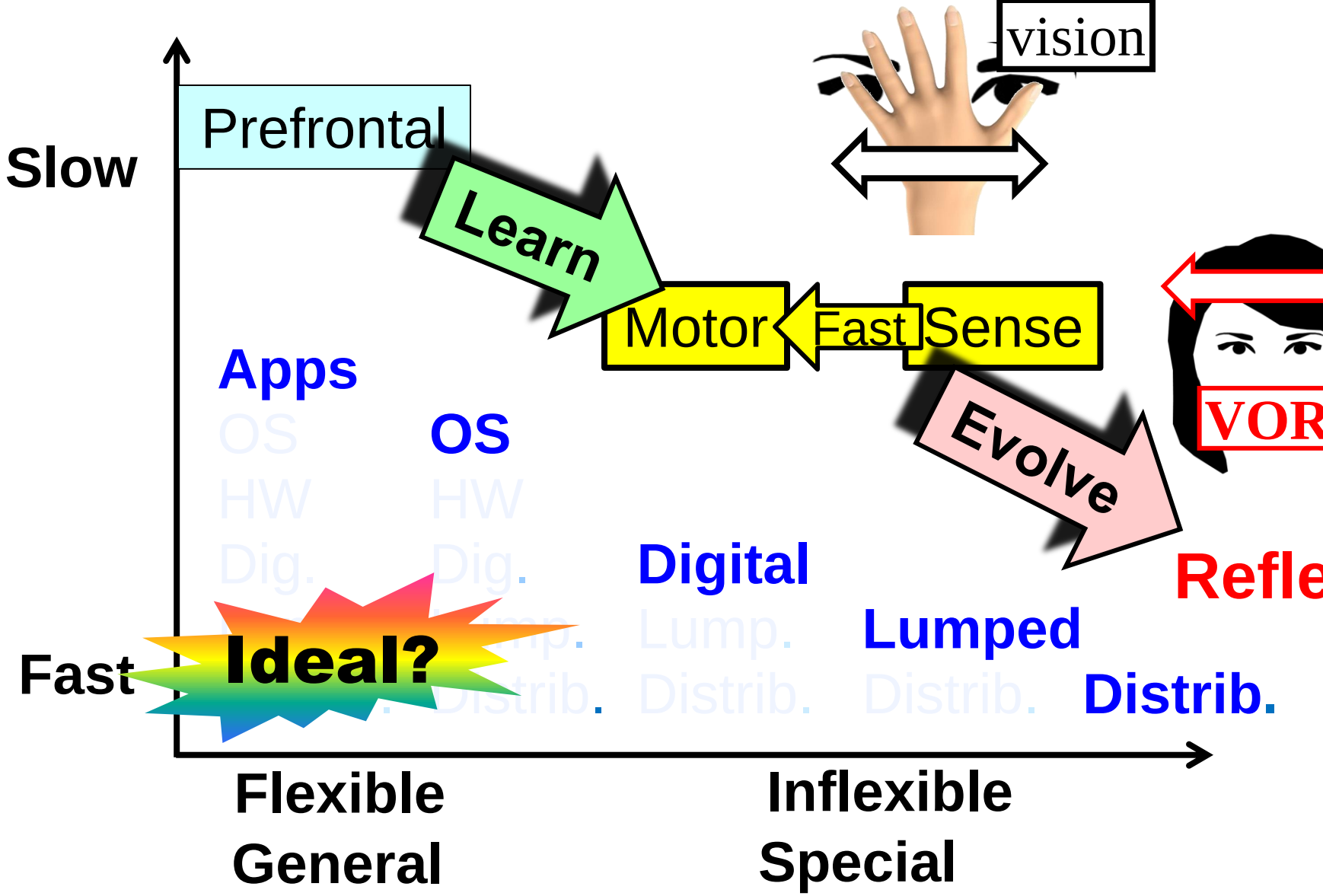
Signal

Analog

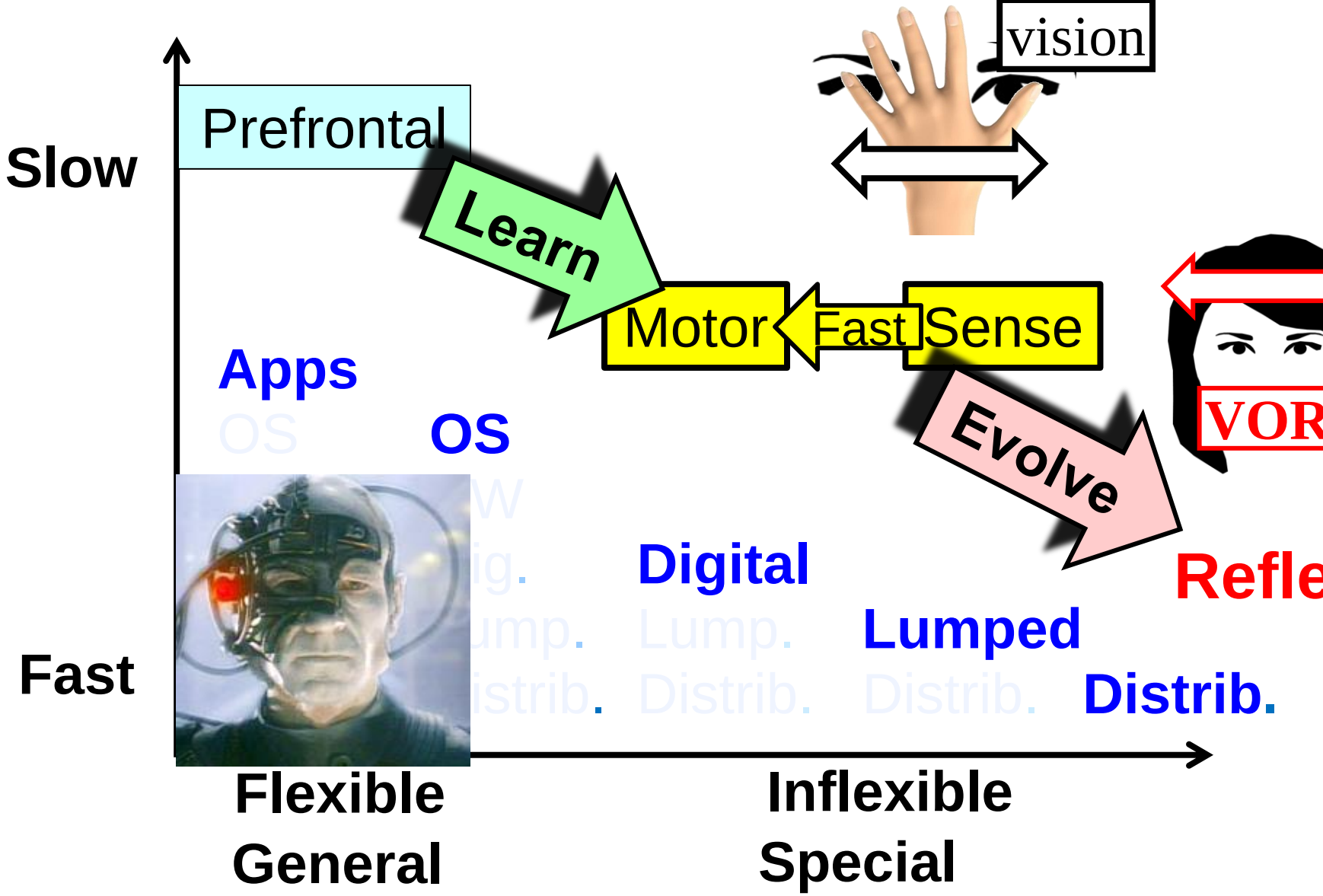
Lumped

Distrib.

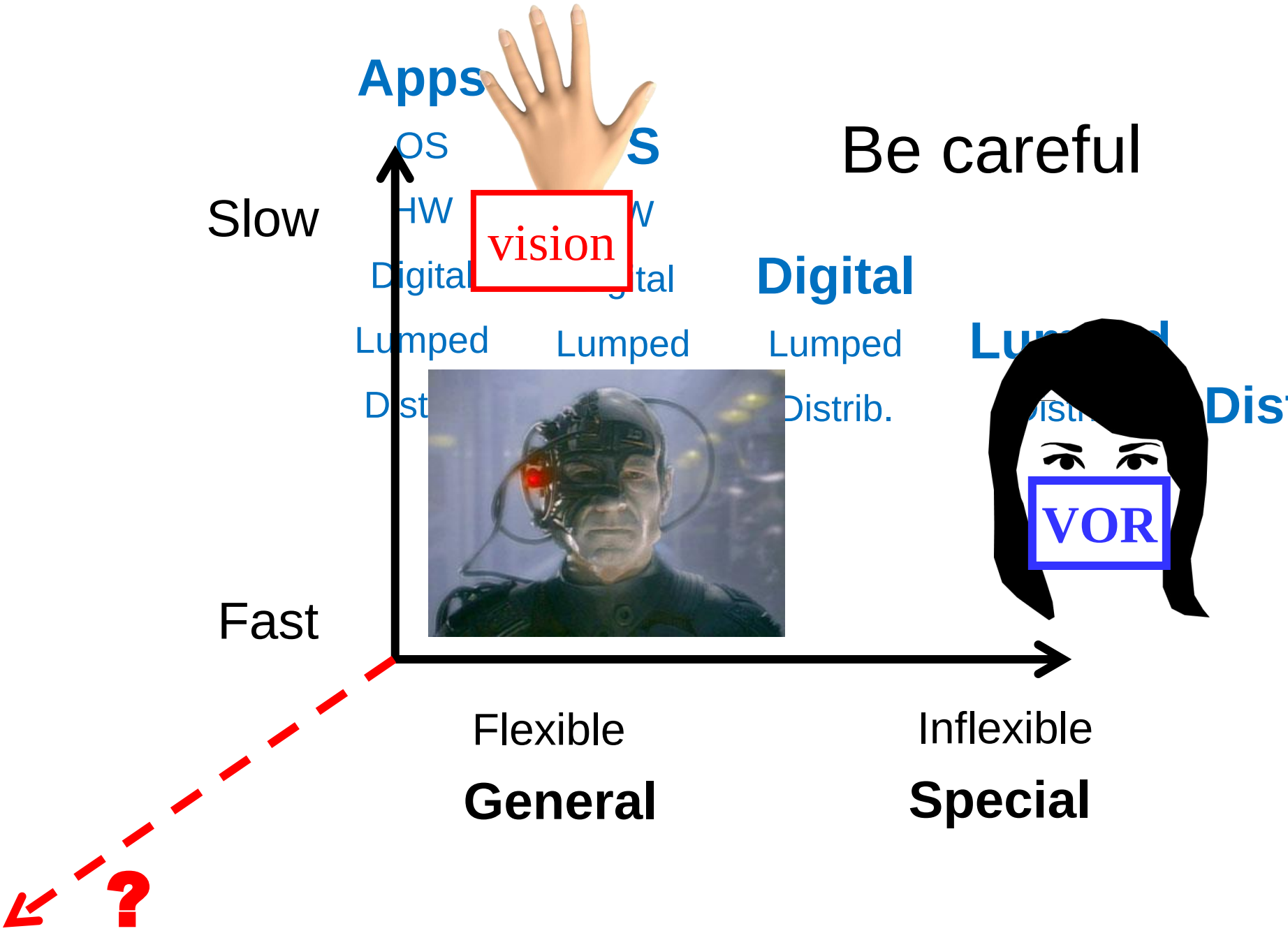
Distrib.

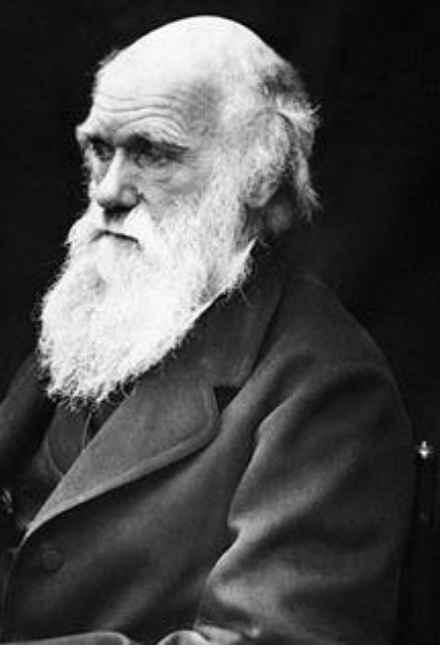


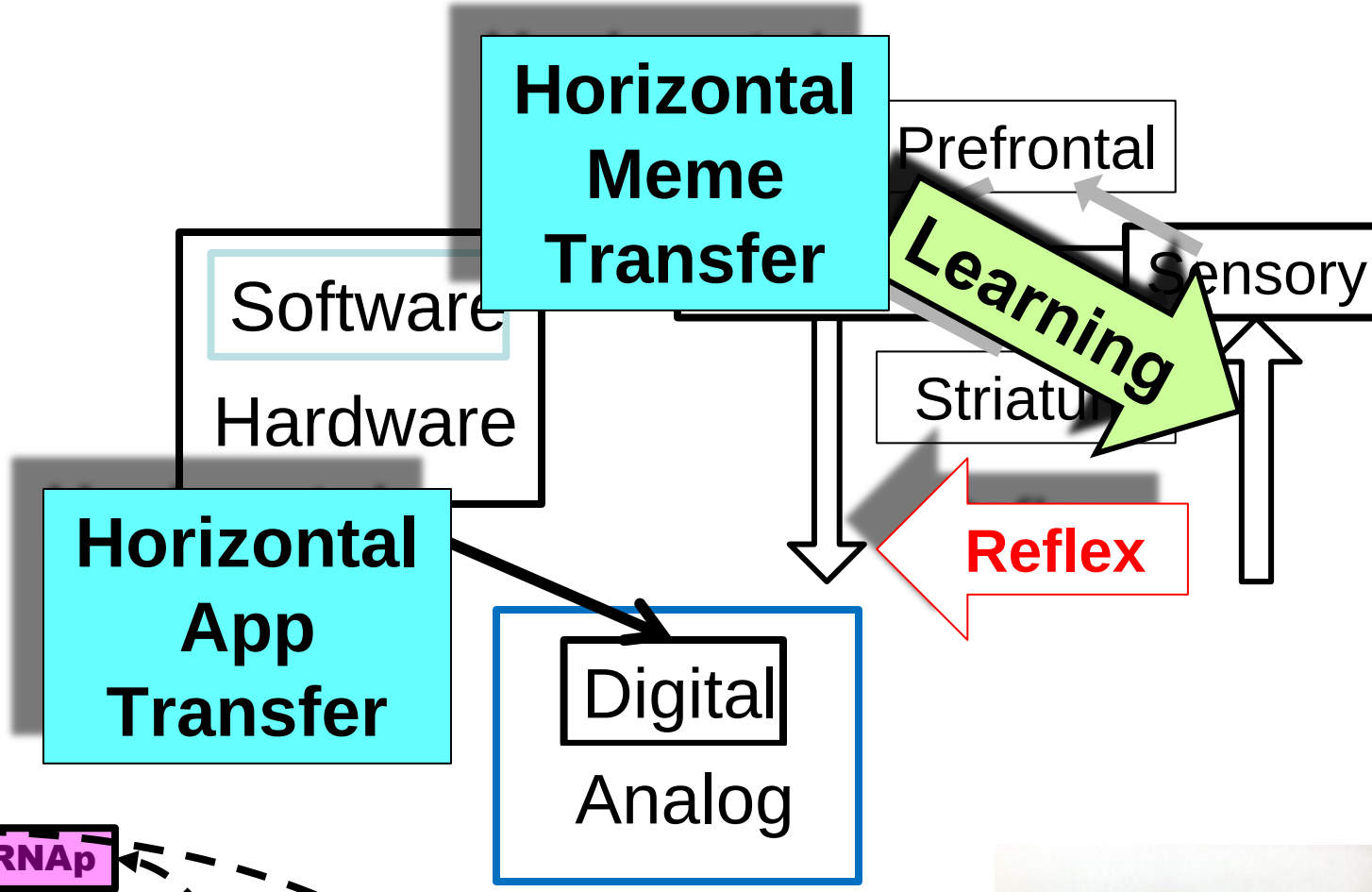
Can a new architecture beat the tradeoff?

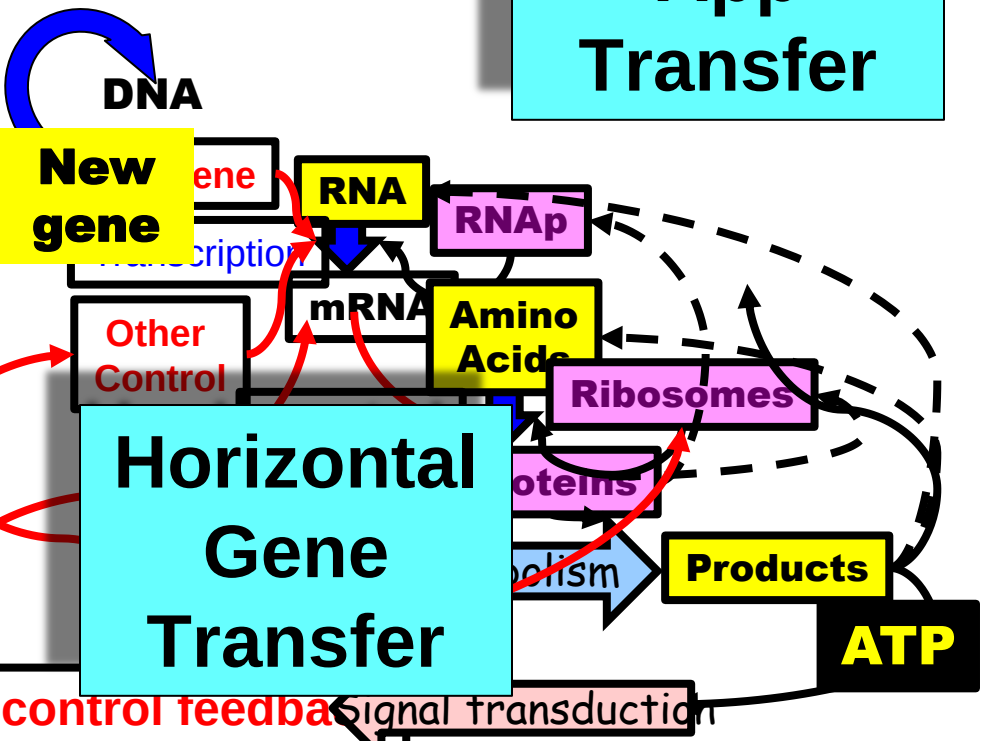


Be careful what you wish for…


Apps
OS
HW
Digital
Lumped
Dist
vision
S
W
tal
Lumped
Be careful
Digital
Lumped
Distrib.
Lu
Dis
VOR
Slow
Fast
Flexible
General
Inflexible
Special
?



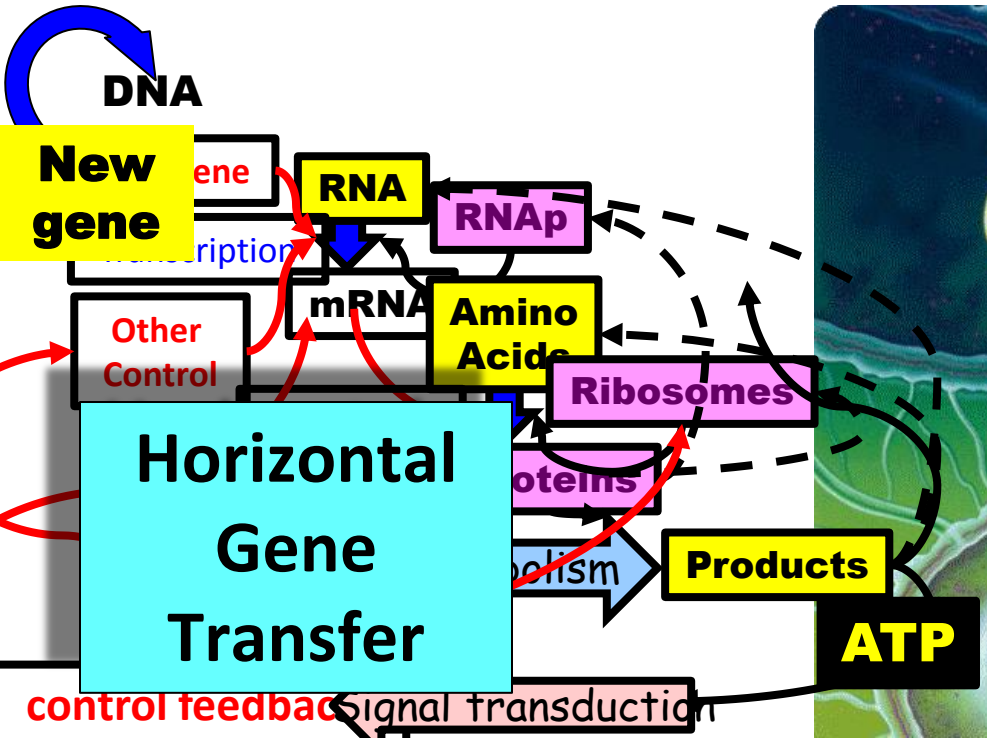

Horizontal Meme Transfer
Prefrontal
Learning
Sensory
Software
Hardware
Striatum
Reflex
Horizontal App Transfer
Digital
Analog

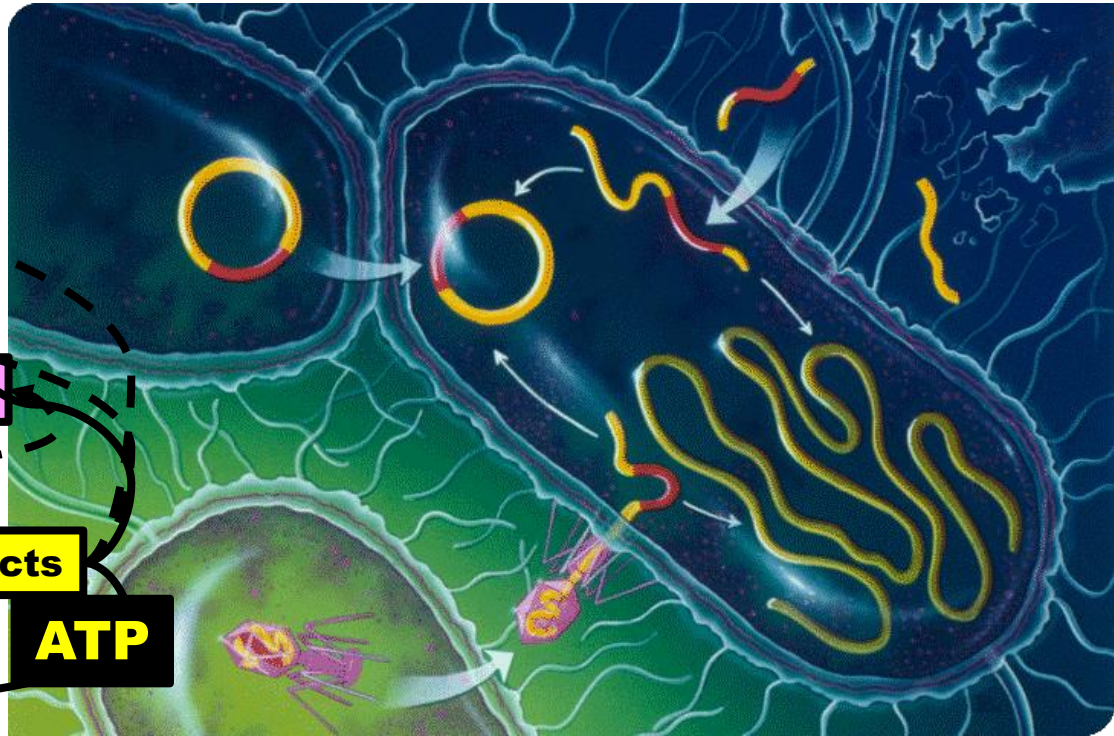

DNA
New gene
RNA
RNAp
mRNA
Other Control
Amino Acids
Ribosomes
Products
ATP
Horizontal Gene Transfer
control feedba
Signal transduction

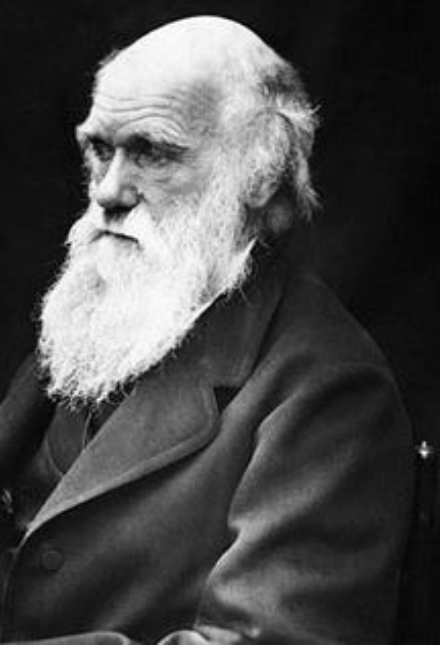
Clinical issue: antibiotic resistance?

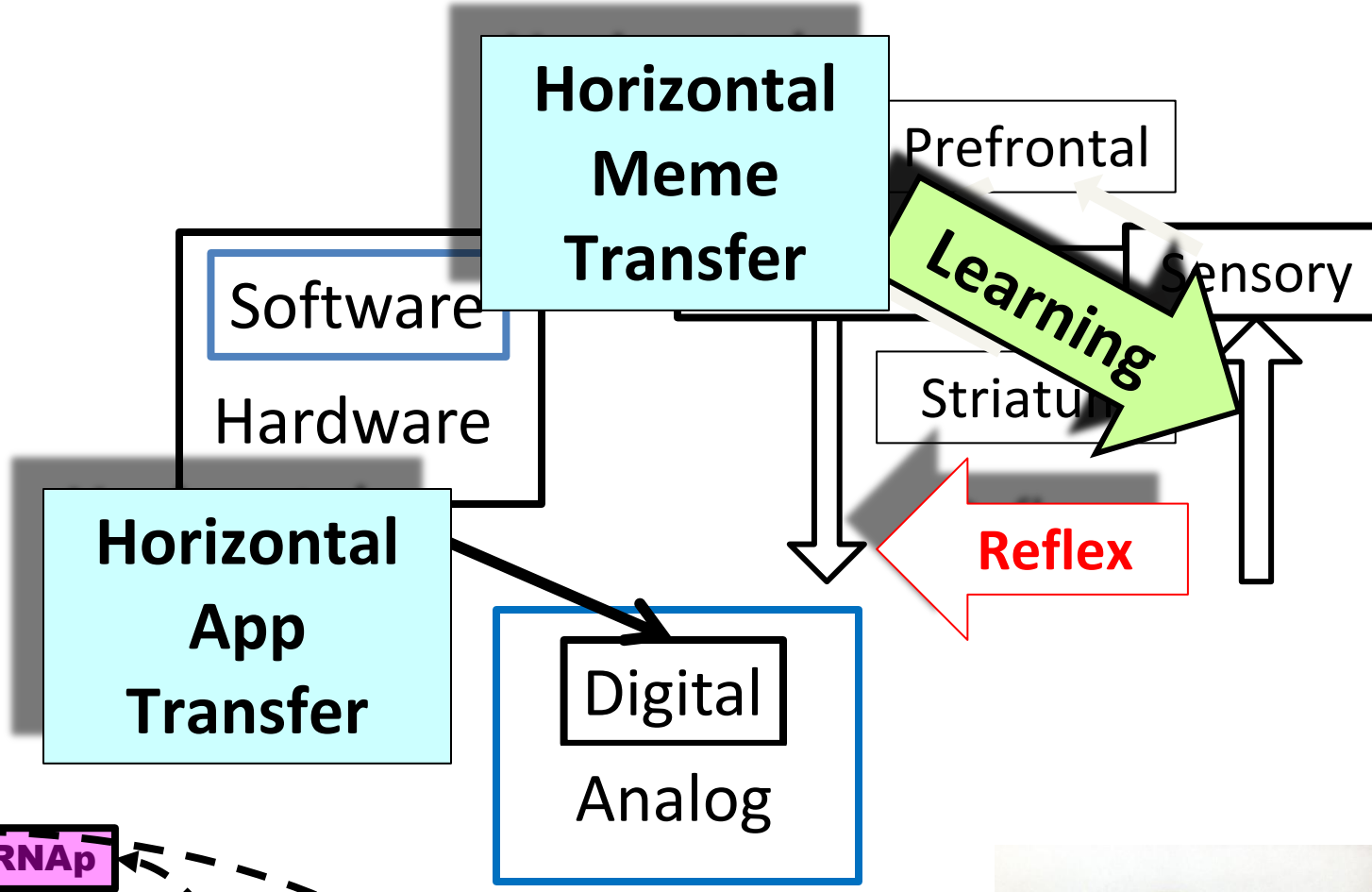
Sequence ~100 E Coli (*not* chosen randomly)

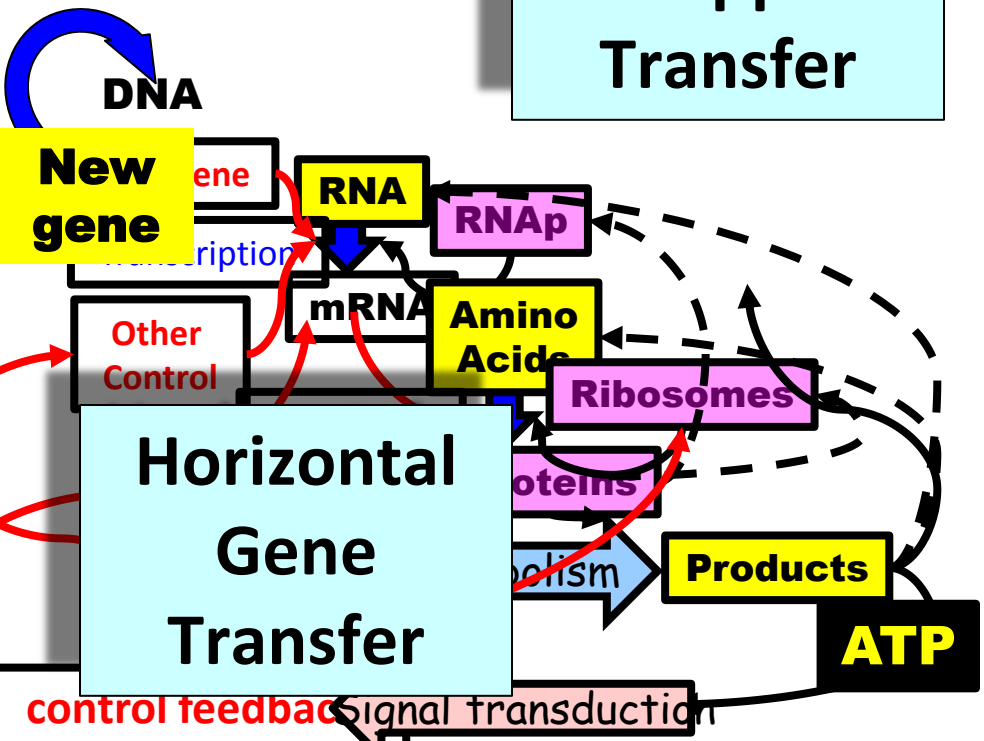
- ~ 4K genes per cell
- ~20K *different* genes in total (pangenome)
- ~ 1K universally shared genes
- ~ 300 essential (minimal) genes








Horizontal Meme Transfer
Prefrontal
Learning
Sensory
Software
Hardware
Striatum
Reflex
Horizontal App Transfer
Digital
Analog


DNA
New gene
RNA
RNAp
mRNA
Other Control
Amino Acids
Ribosomes
Products
ATP
Horizontal Gene Transfer
Signal transduction

Mechanisms for Horizontal Transfer

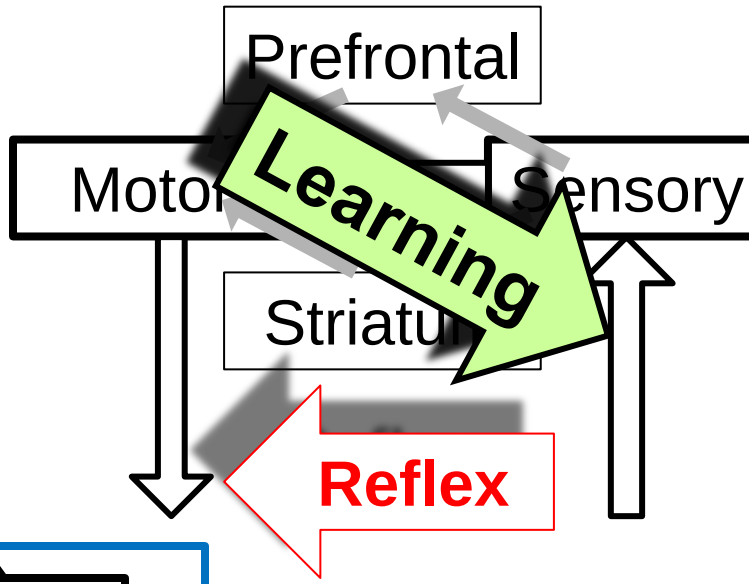
Software

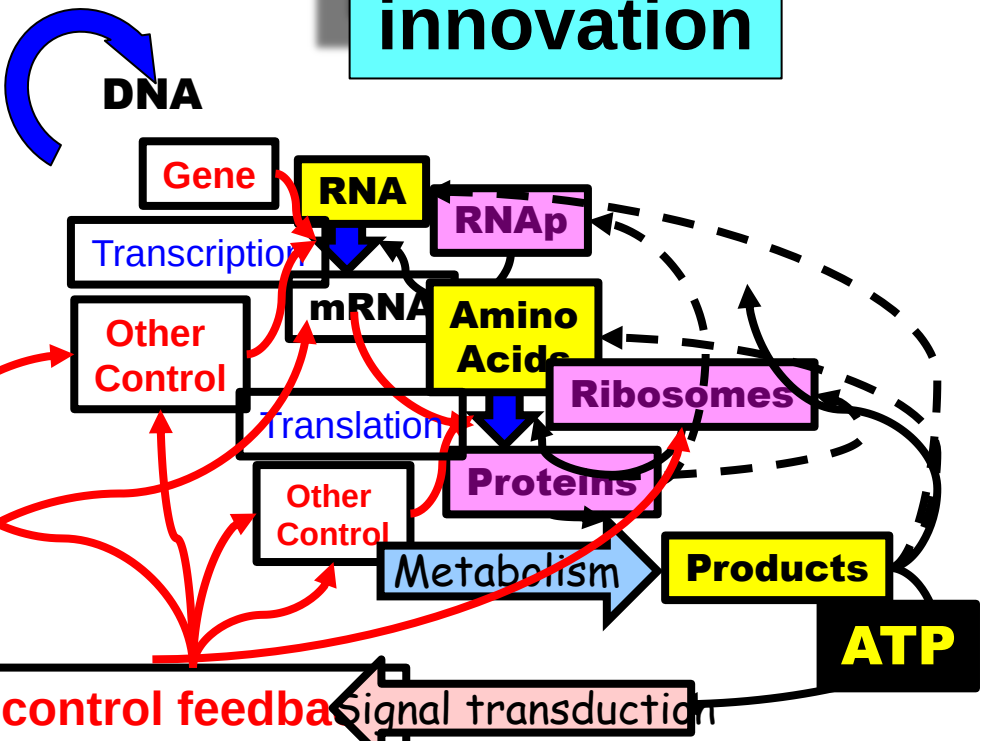
Hardware

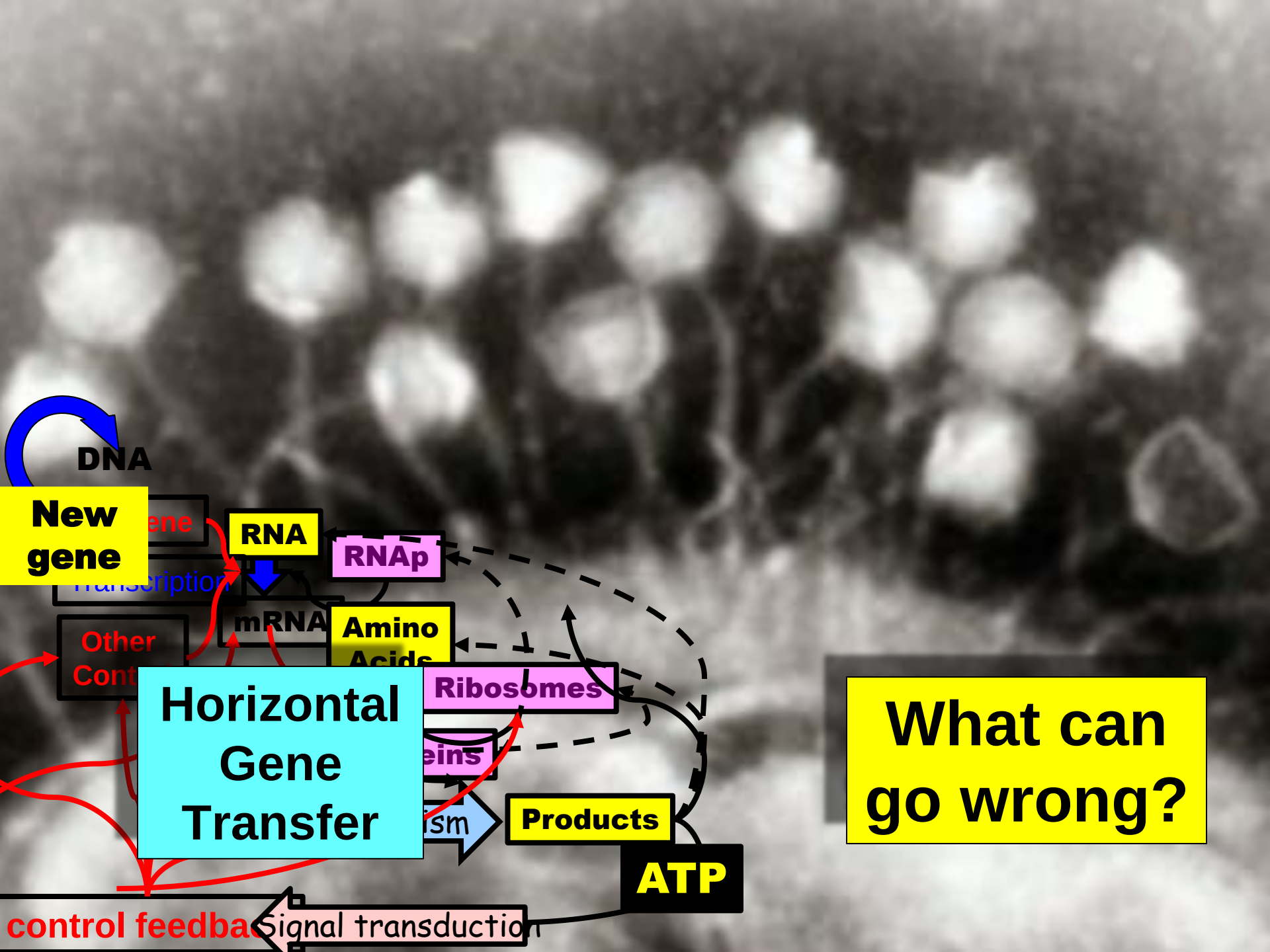
Enable "vertical" innovation

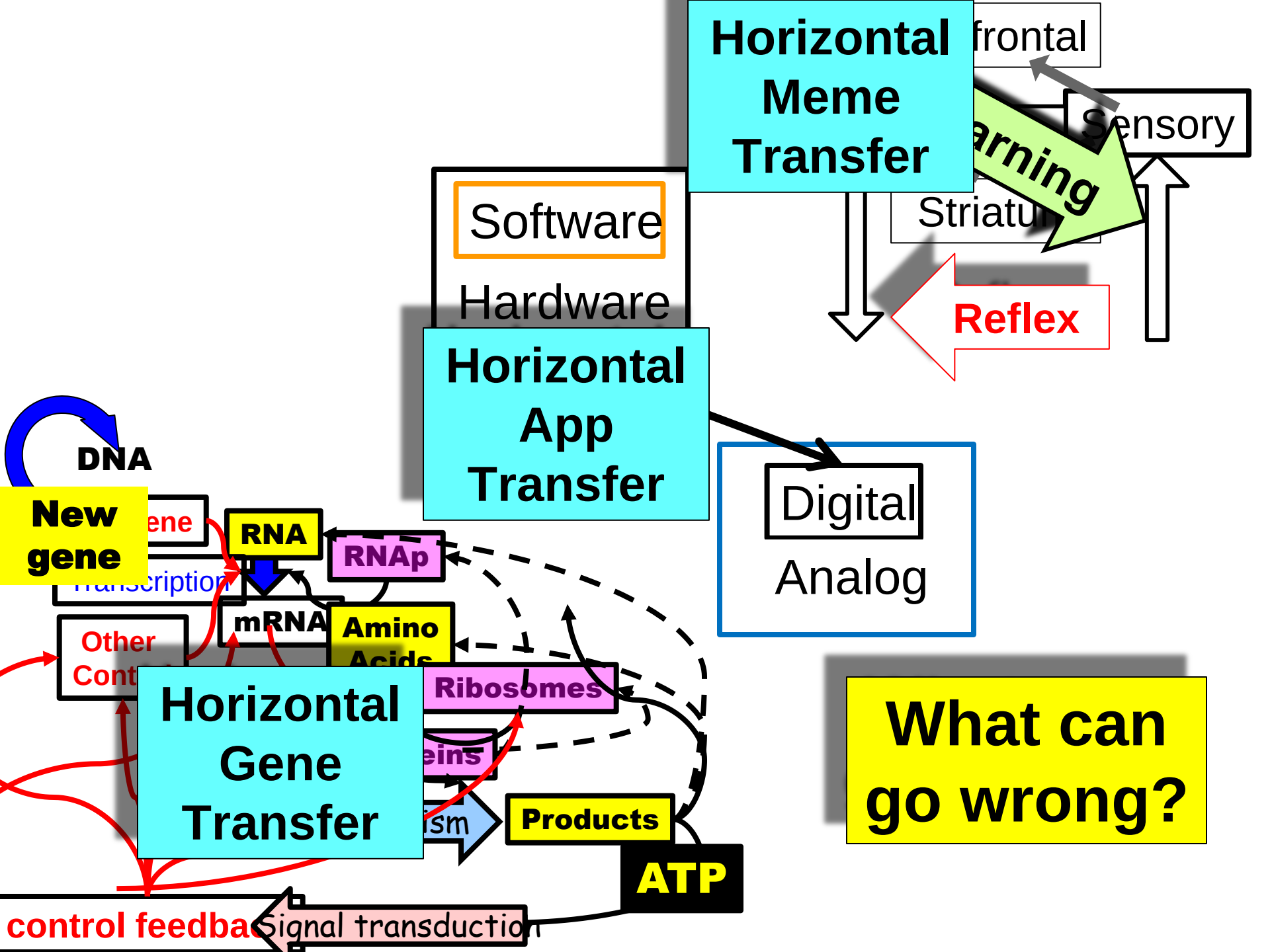
Digital

Analog






DNA
New gene
RNA
RNAp
mRNA
Amino Acids
Other Cont
Ribosomes
Horizontal Gene Transfer
Products
ATP
control feedba
Signal transduction
What can go wrong?


Horizontal Meme Transfer
frontal
Sensory
arning
Striatu
Software
Hardware
Reflex
Horizontal App Transfer
Digital
Analog
DNA
New gene
ene
RNA
RNAp
Transcription
mRNA
Amino Acids
Other Cont
Ribosomes
eins
sm
Products
ATP
Horizontal Gene Transfer
What can go wrong?
control feedba
Signal transduction

Exploiting layered architecture

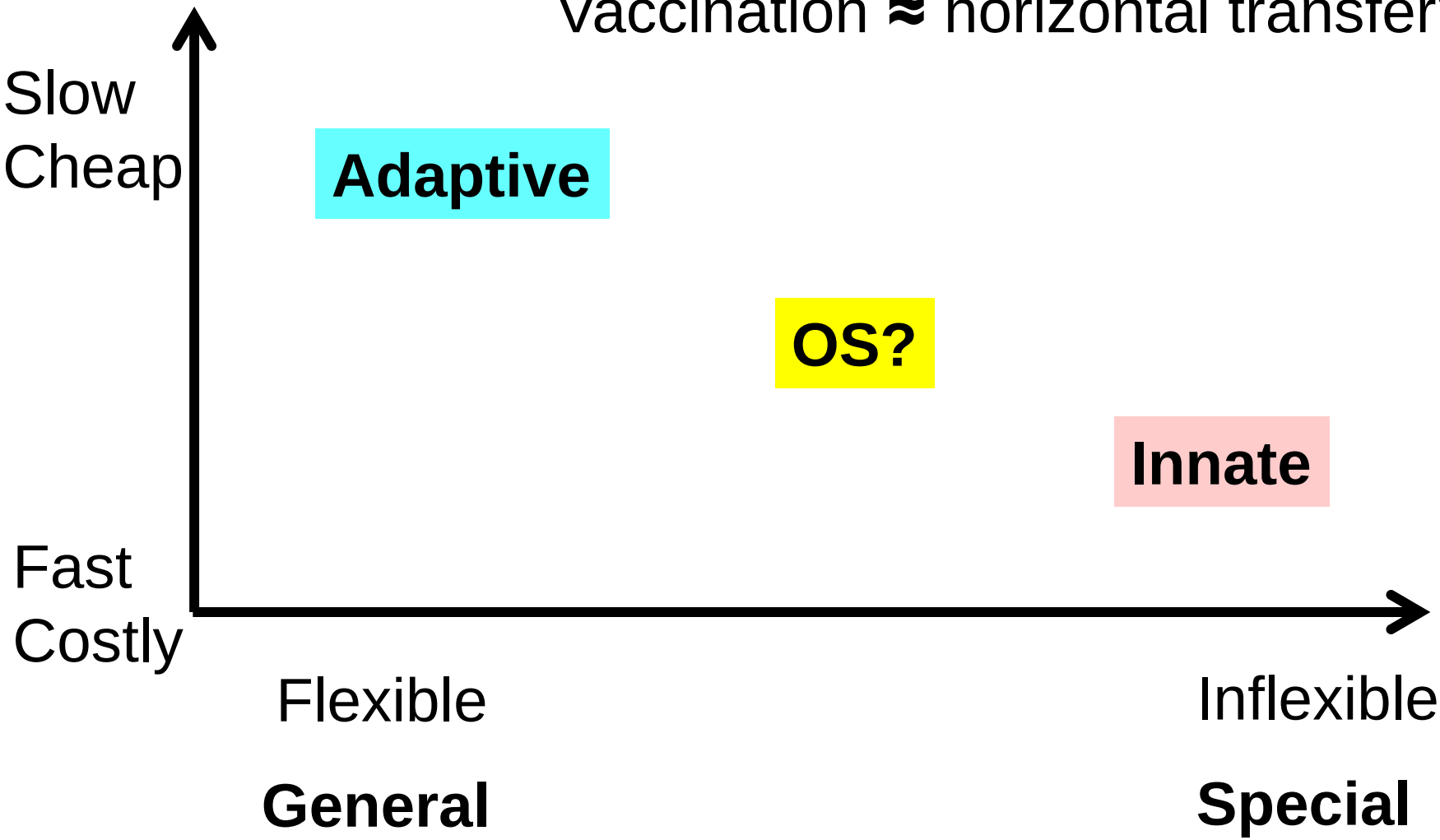
Meme → Horizontal **Bad Meme** Transfer

Virus → Horizontal **Bad App** Transfer

Fragility?

Horizontal **Bad Gene** Transfer ← **Virus**

Parasites & Hijacking

Layered immunity

Vaccination ≈ horizontal transfer?