Recovering from System Failure

Andriy Drozdyuk

Outline

1. Types of bugs
2. Restart only software
3. Fail fast
4. The Happy Path
5. Error checking
6. External System Failure
7. Long running processes
8. Questions?
9. Conclusion: Apollo Software

Total time: 48 minutes

"it is correct to say that we landed on the moon with 152 Kbytes of computer memory."

—Don Eyles, Apollo 11 programmer

Types of bugs

Bohrbugs

Solid

Easily repeatable

Observable

Heisenbugs

Bohrbugs

Solid

Easily repeatable

Observable

Heisenbugs

Unreliable

Occur only under certain conditions

Observation can alter conditions

Bohrbugs **Heisenbugs**

Frequent

Development

Bohrbugs	Heisenbugs
Frequent	Infrequent
	1 in 1,000,000,000 requests

Development

Bohrbugs

Heisenbugs

Never in core

Often in ancillary

Production

Bohrbugs

Never in core

Often in ancillary

Heisenbugs

All the time!

100k req/sec is a bug every 3 hours

Production

Bohrbugs

Easy

Heisenbugs

Hard

Testing

Most bugs are…?

Most bugs are Heisenbugs!

131/132 errors are transient

“Why Do Computers Stop and What Can Be Done About It?” by Jim Gray, 1985

“Now the computer issued code 500. It thought the landing radar antenna was in the wrong position.”

—Tales from the Lunar Module Guidance Computer

Restart only software

Why restarting works?

In Bohrbugs?

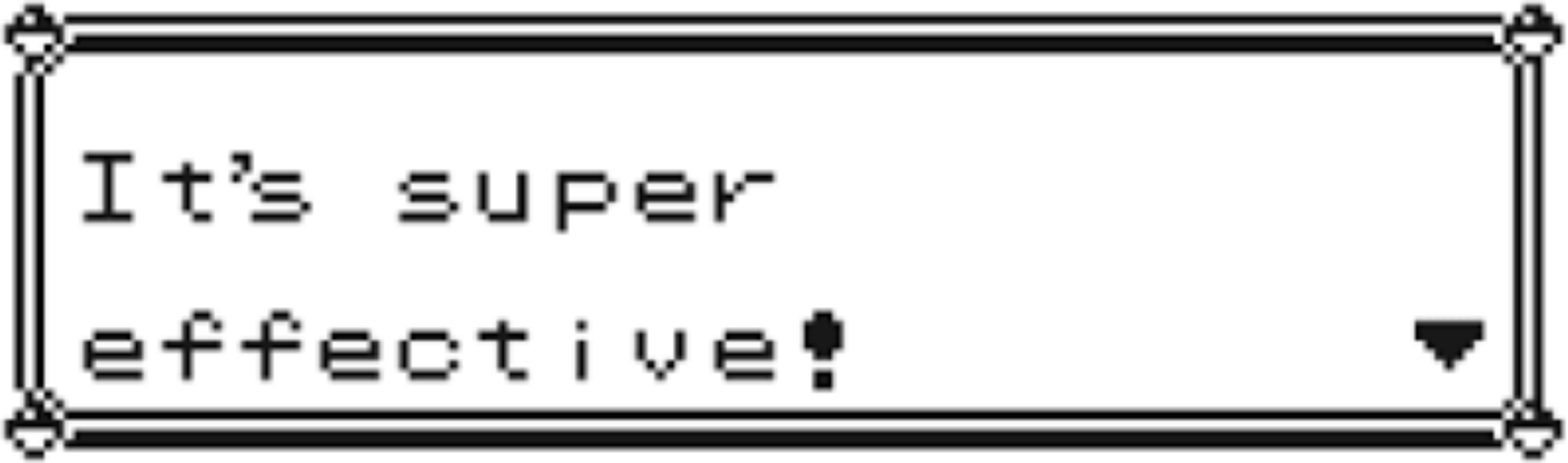
Core

Ancillary

In Bohrbugs?

Core	Nope. Same error again.
Ancillary	Sometimes. E.g. if you don’t care about site counter failing.

In Heisenbugs?



It's super
effective! ▼

In Heisenbugs?

Restarting makes them disappear

“Then we heard the words ‘program alarm’.”

Larson gave thumbs-up. (He later said he was too scared to form words.)

“Again a program alarm light.”

Again “go” from the ground.

Fail fast

How to fail fast?

How to fail fast?

Do not try to recover

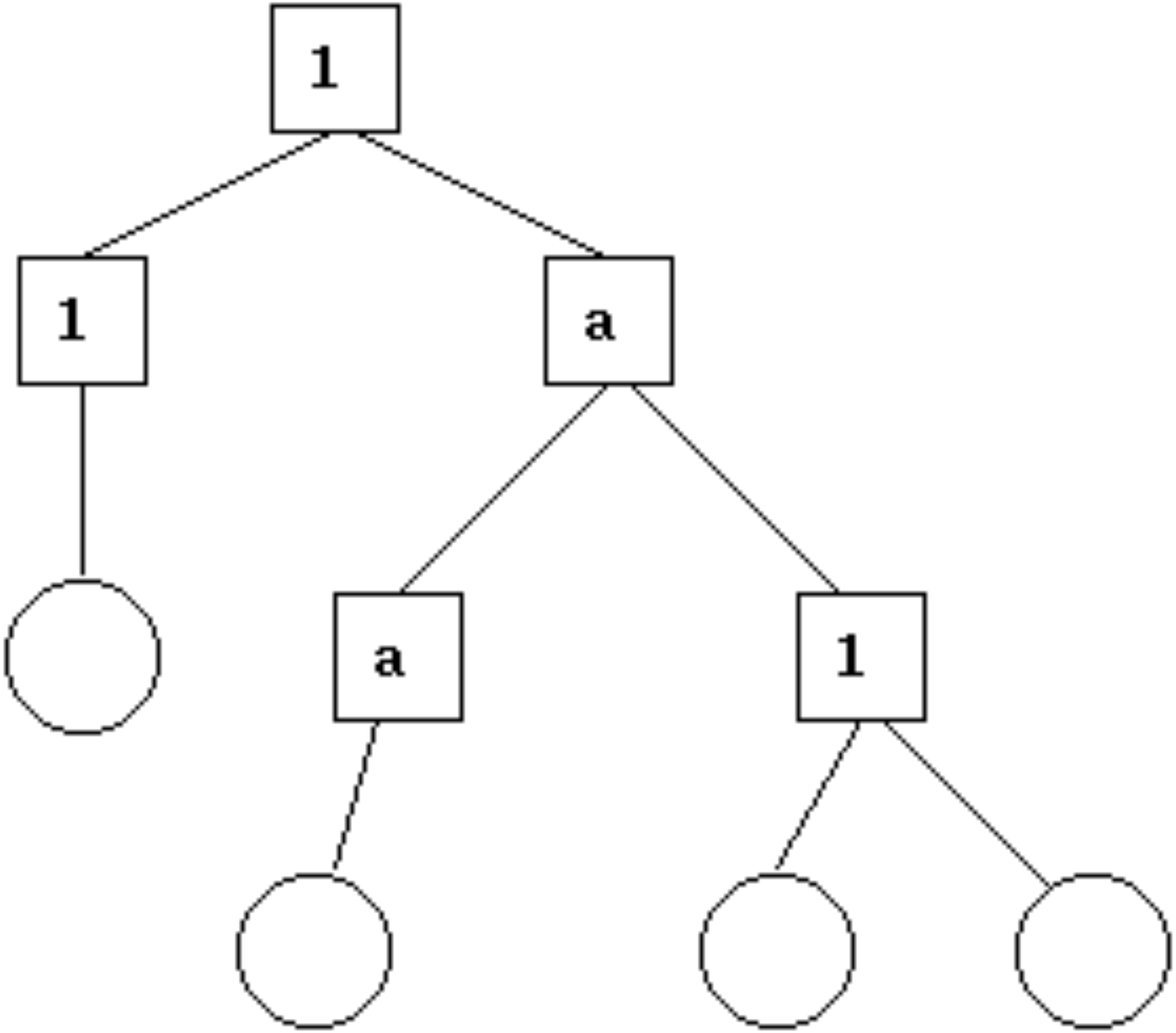
Do not fix the error

Just crash

Who handles the failure?

Someone else!

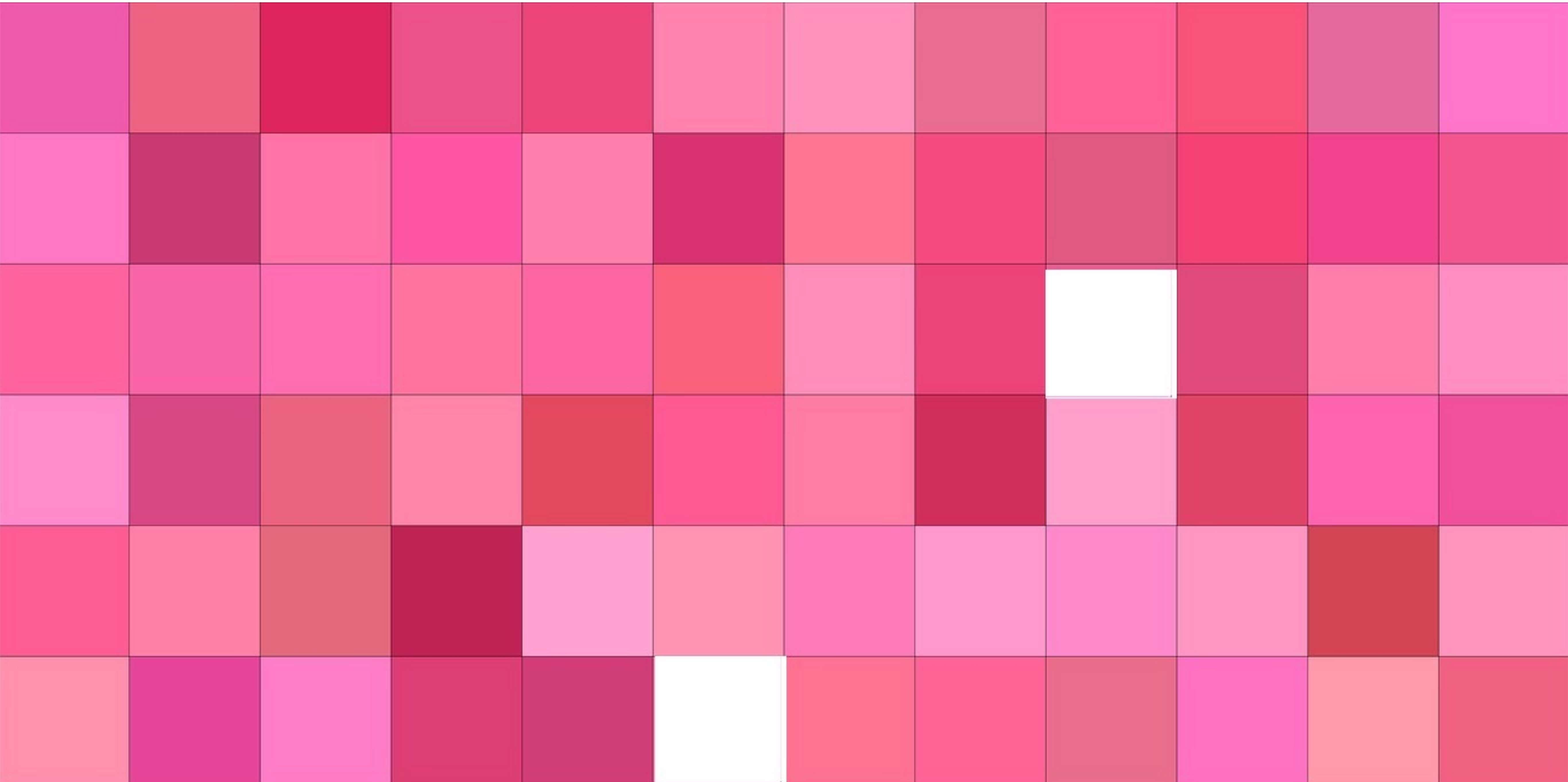
Introduce supervision



Error Kernel

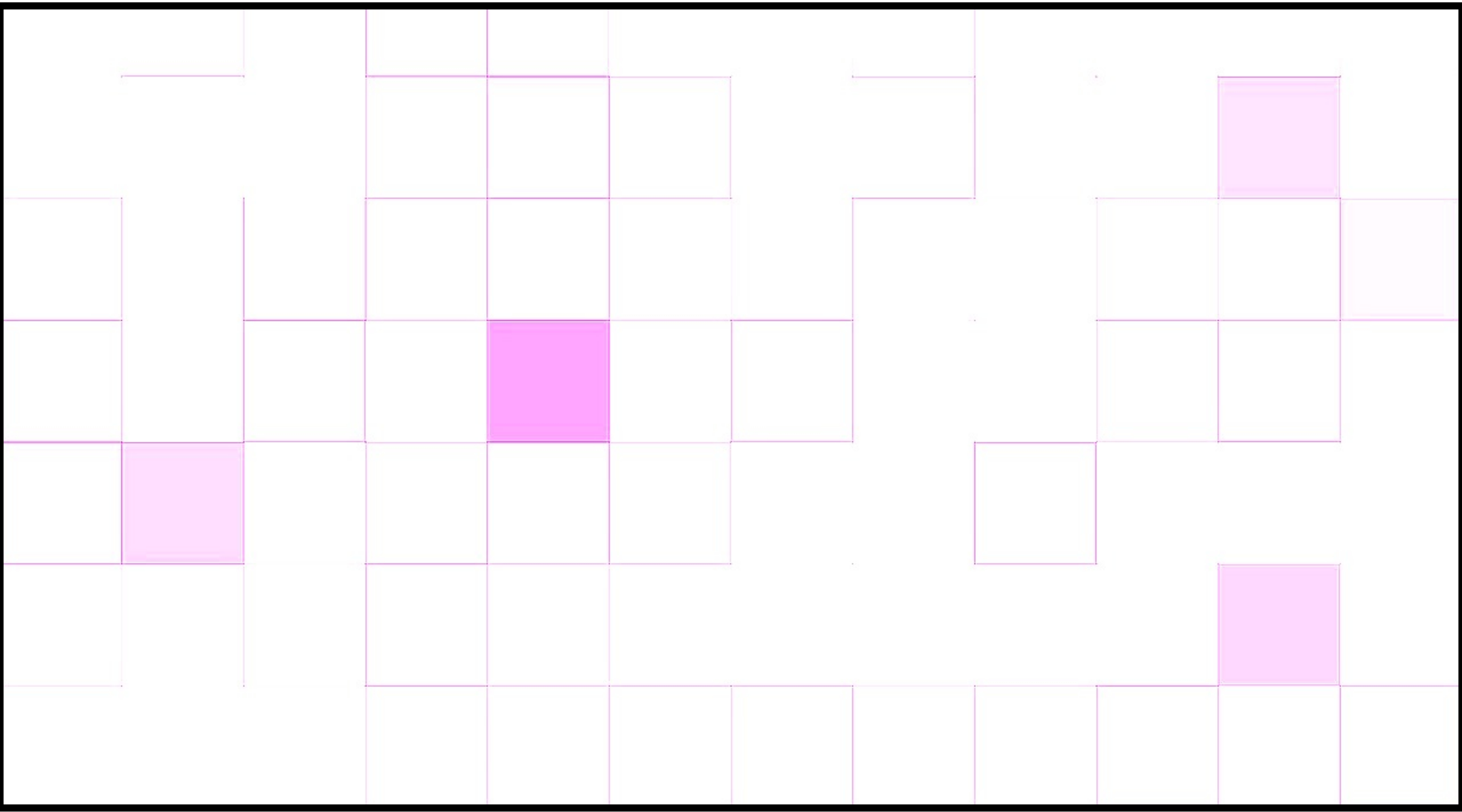
Part of the program that has to be correct.

Error Kernels typical software

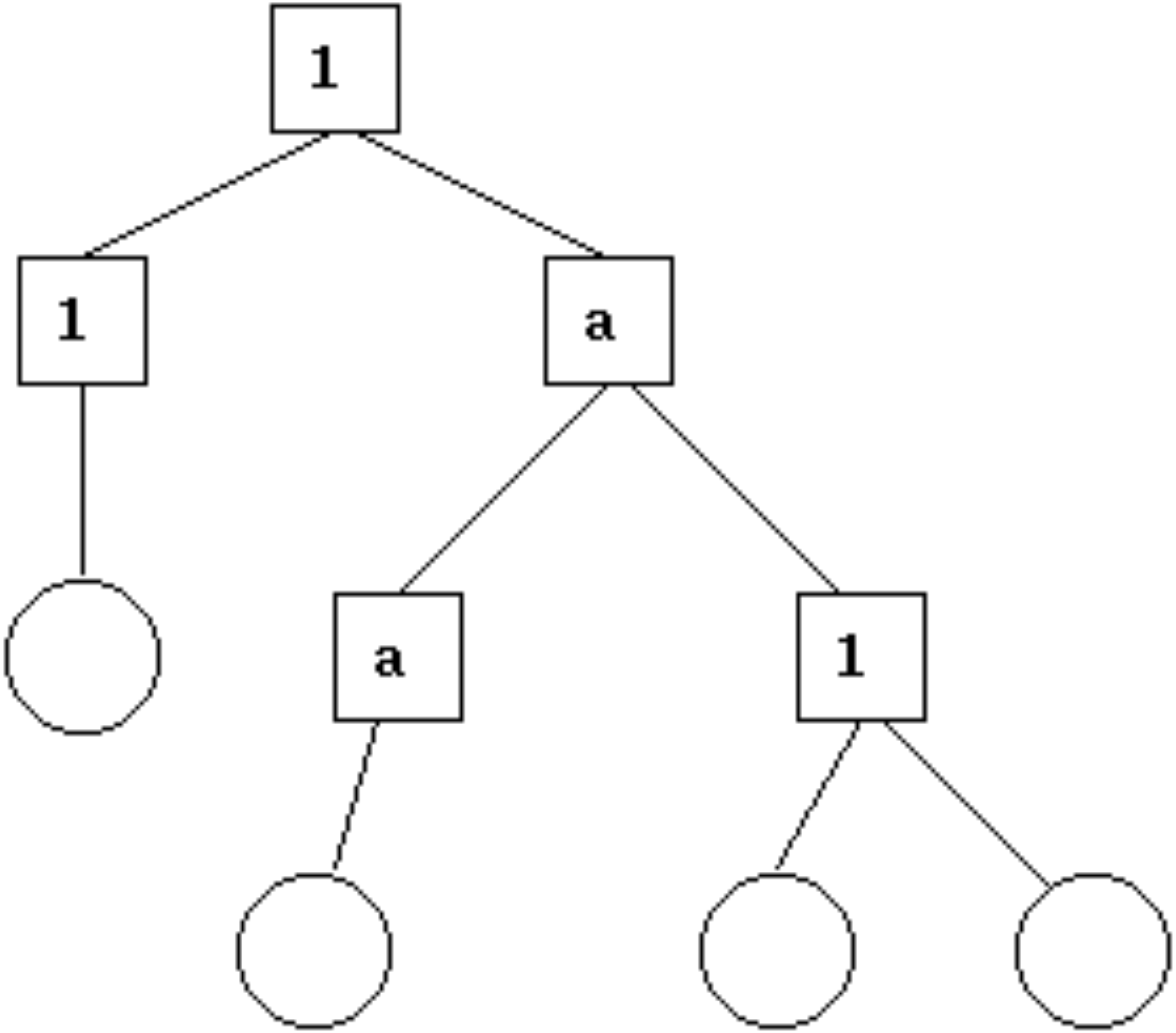


Error Kernels

desired

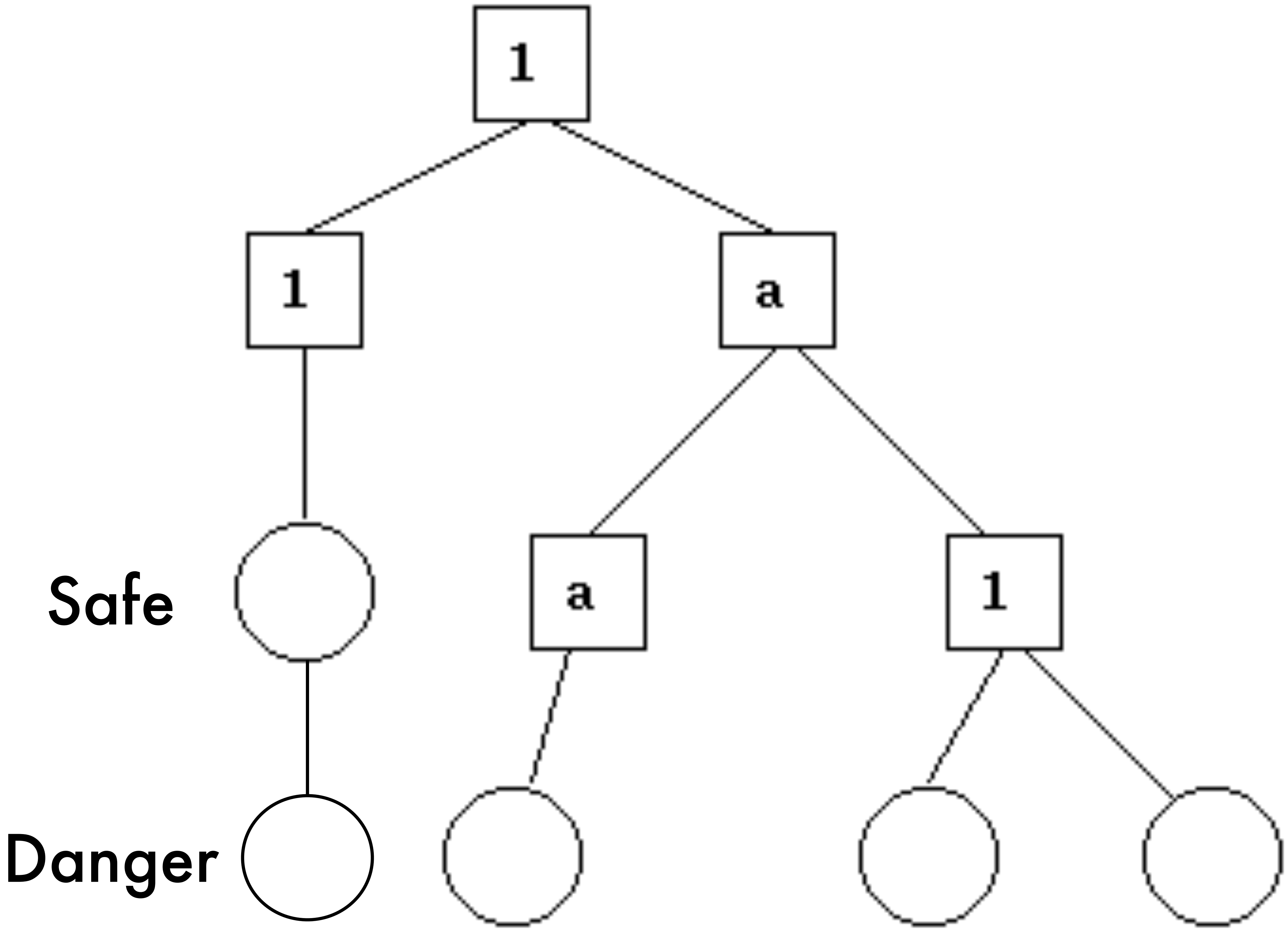


How?



Danger

Push danger down



“Ah! Throttle down... better than the simulator” commented Aldrin, “Throttle down on time!” exclaimed Armstrong

In the official transcript of communications (...) these are the only exclamation points.

Happy path

Happy path

The environment is as you expect it to be.

Happy path

Program for normal case only

Don’t try to fix and continue

When failing - log

At MET 102:42:17 a 1201 alarm occurred.

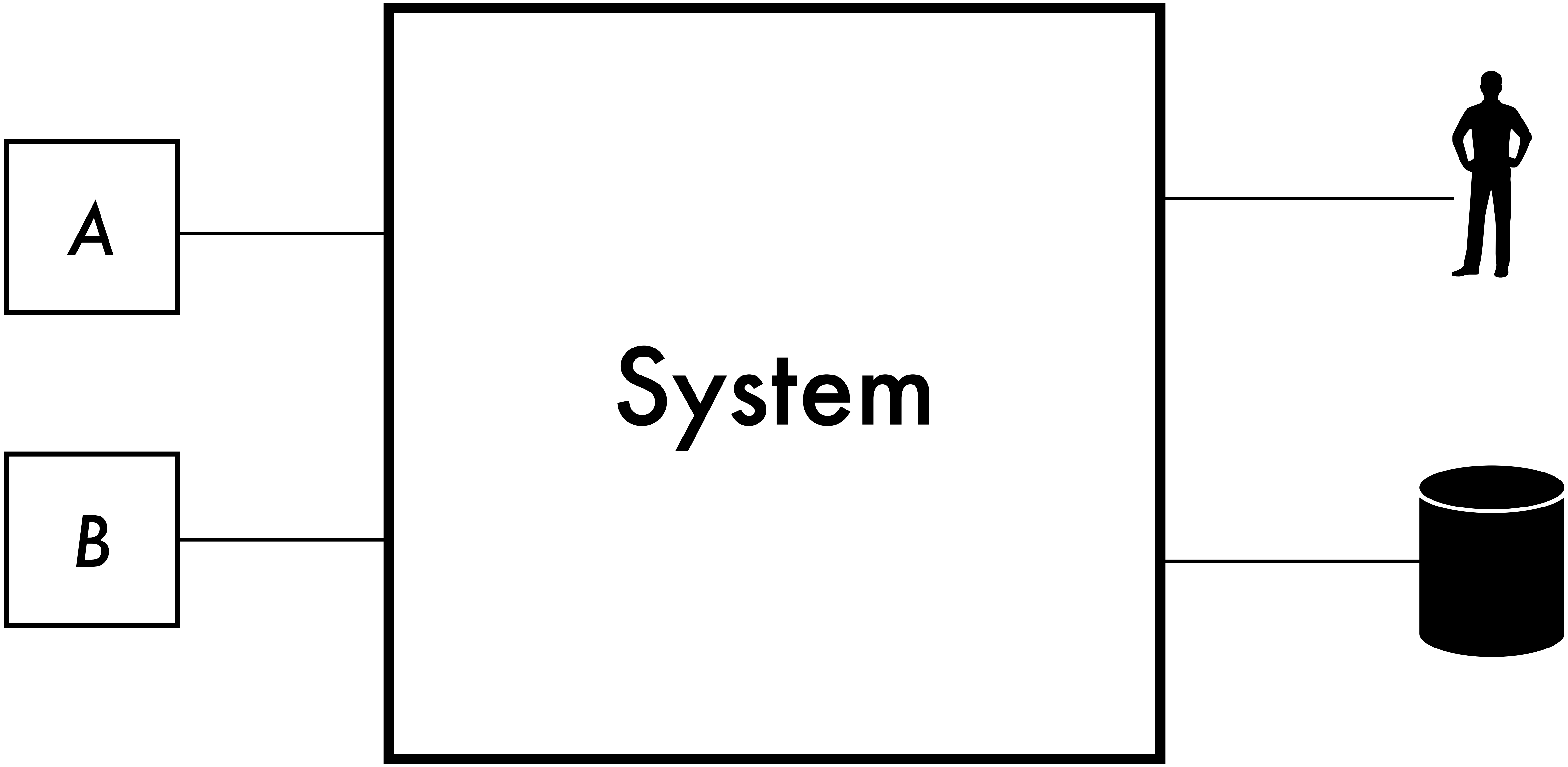
24 seconds later there was another 1202.

Just 16 seconds later (...) yet another 1202

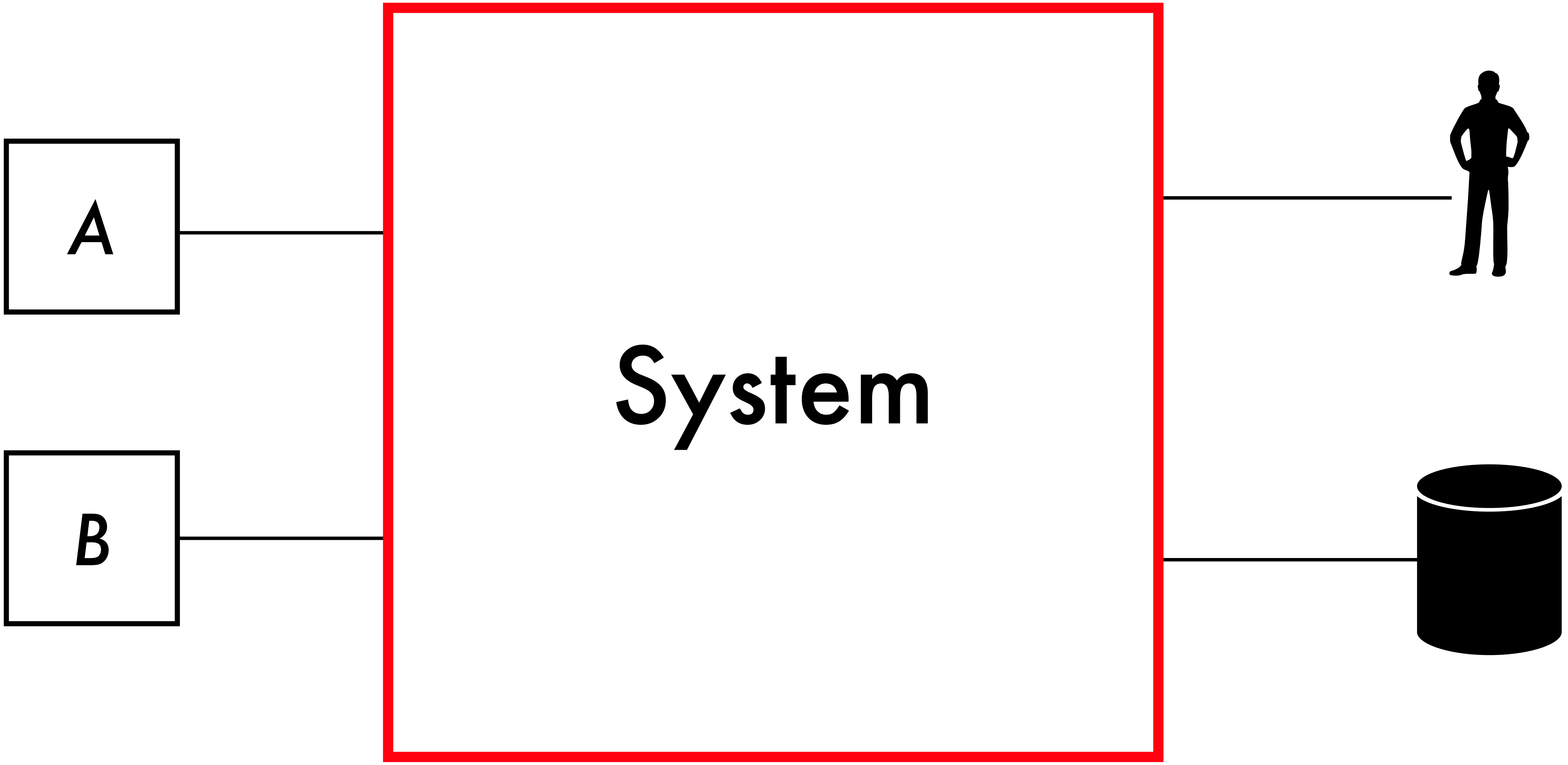
Mission control in Houston
called a "go" in each case.

Error checking

Where to check for errors?



System boundary



Exception handling

Exception handling

Don’t

Exception handling Don’t

1. Most examples are bad

2. You can’t do anything about them

3. Hides the error from top level processes

Exception handling Hacks

1. Declare method throws

2. Convert to runtime

Neil Armstrong, whose heart rate rose from 120 to 150 during this period, put it this way:

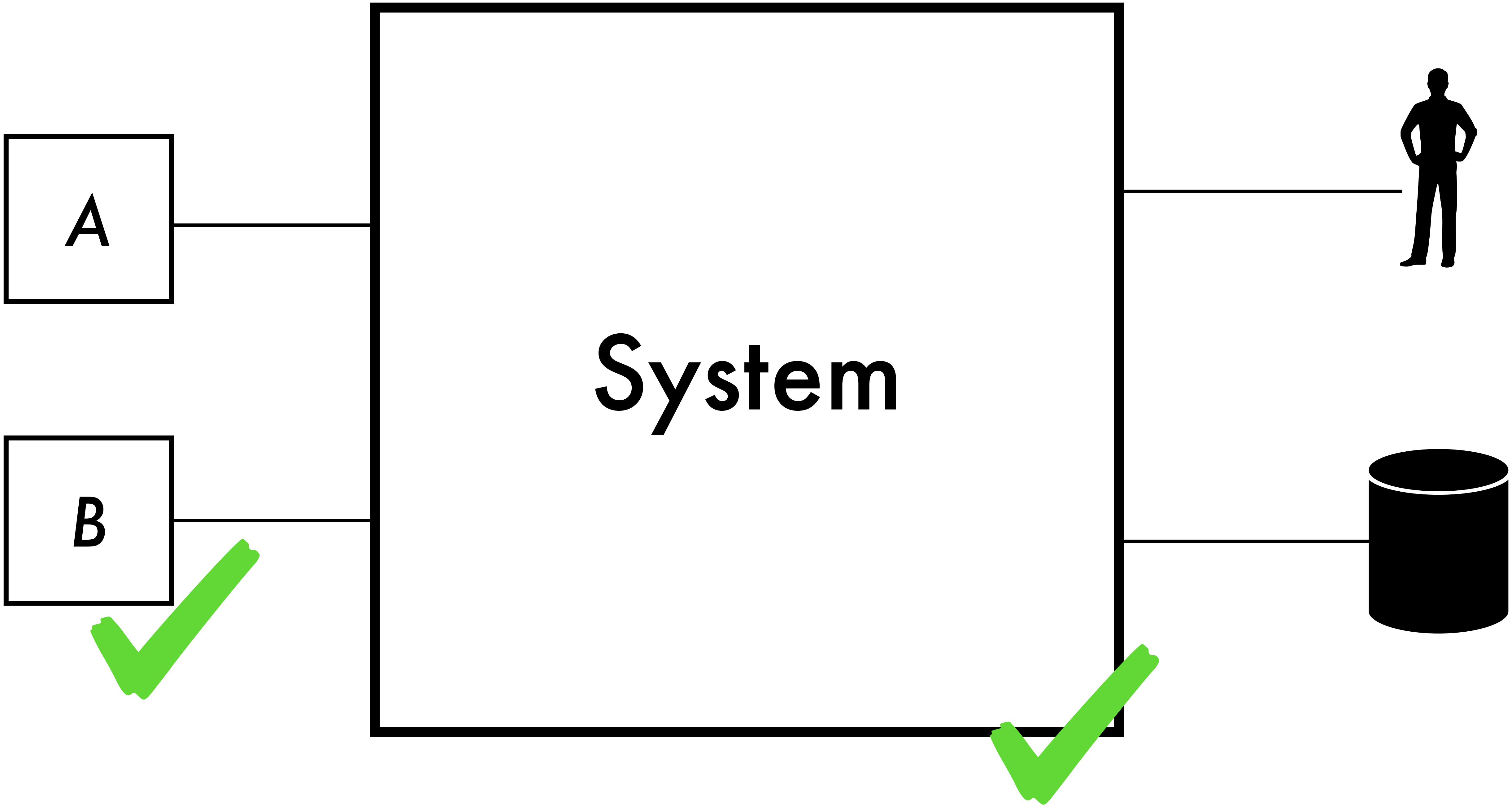
> "...concern here was ... whether we could continue at all."

External System Failure

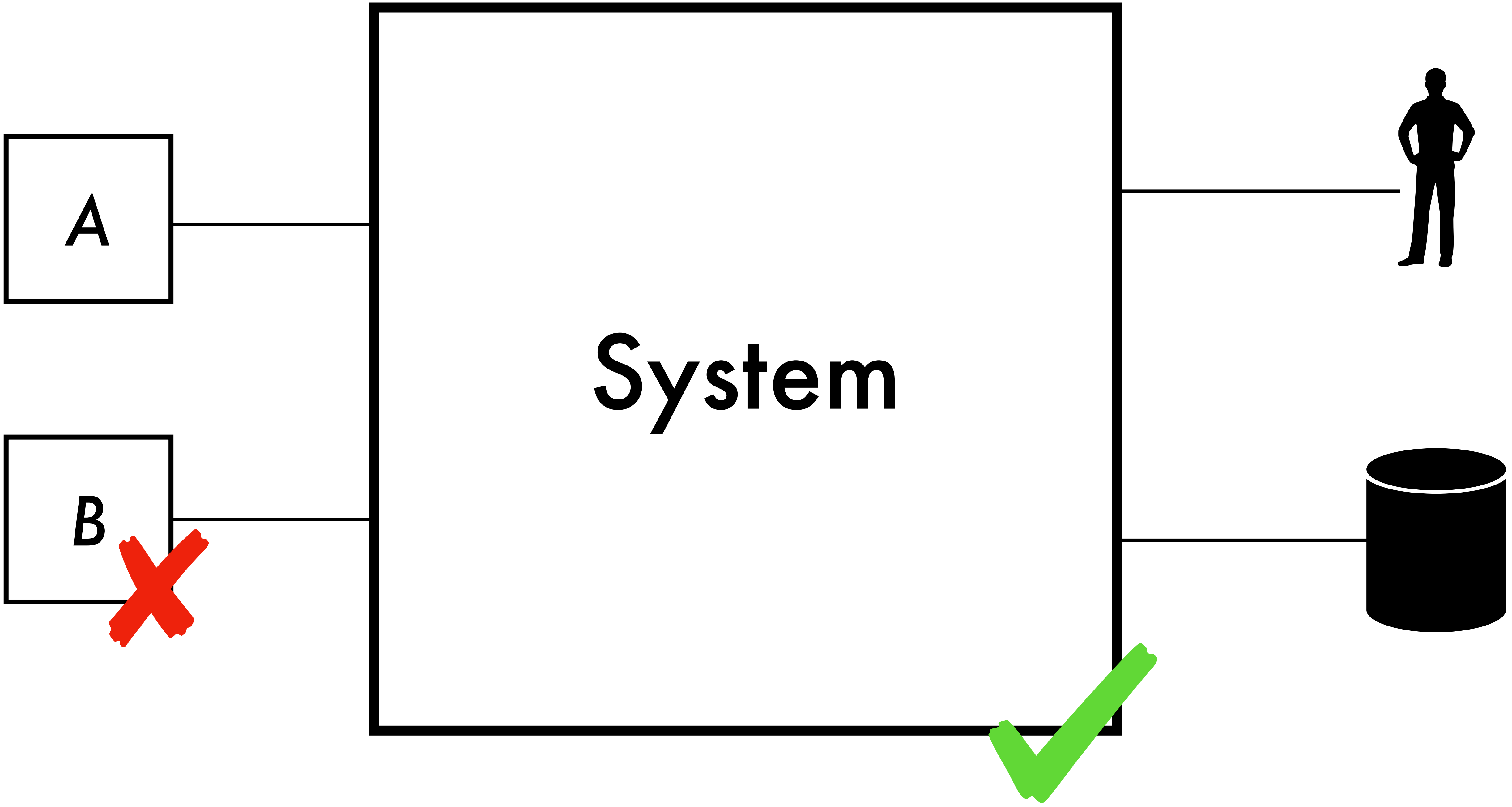
External system failure

1. Crash
2. Restart
3. Wait for external systems
4. Continue

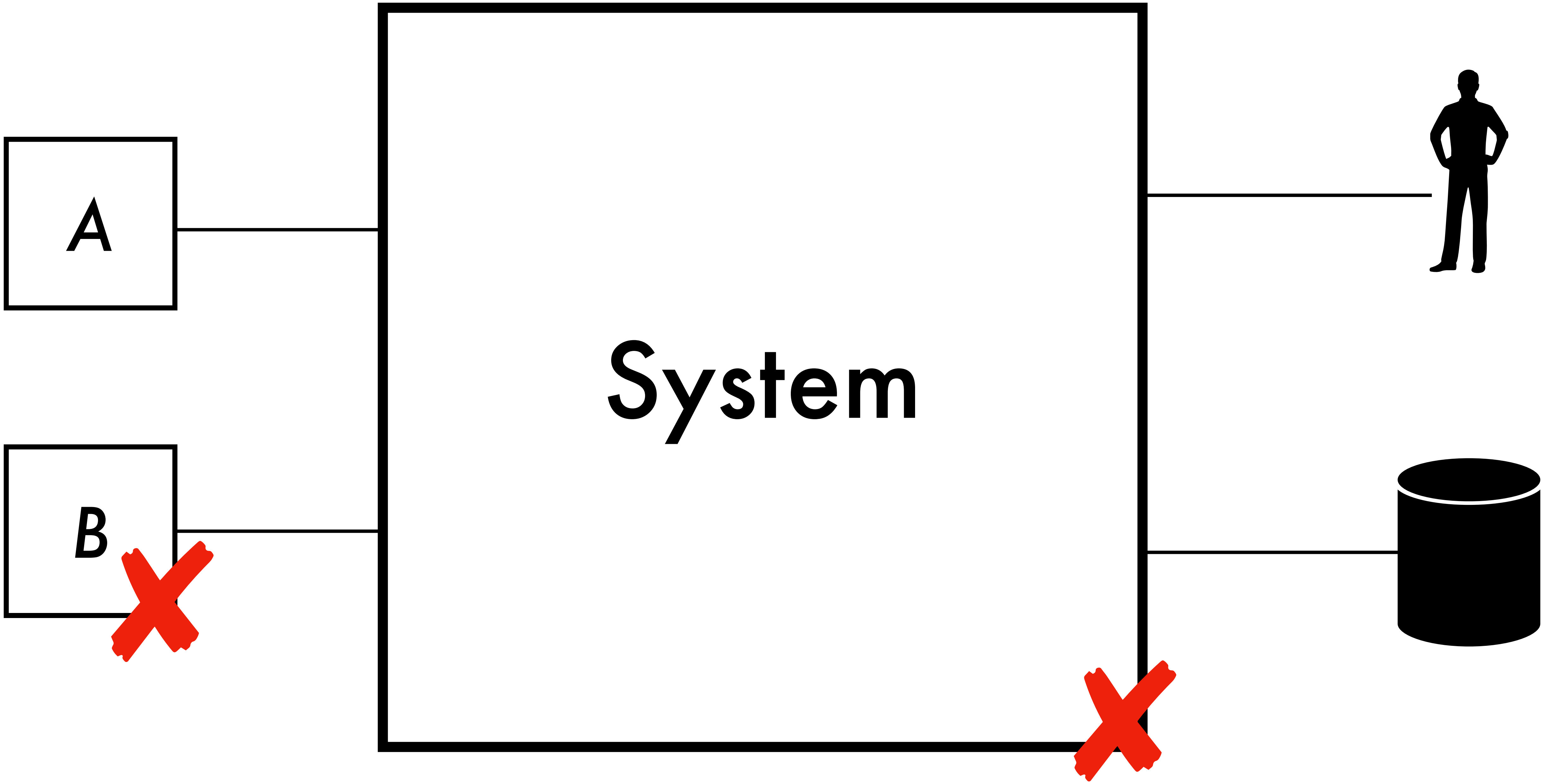
Normal



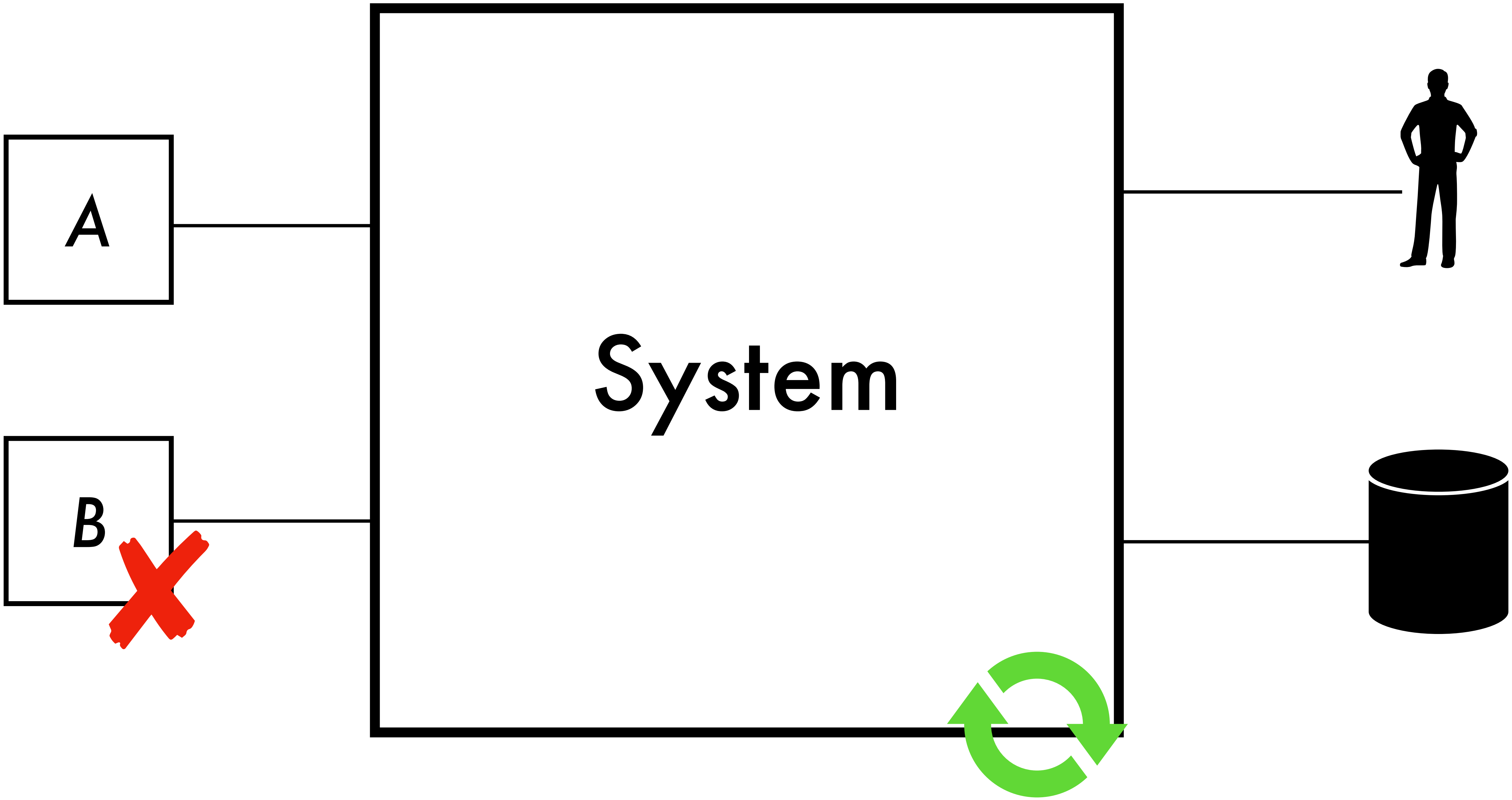
External failure



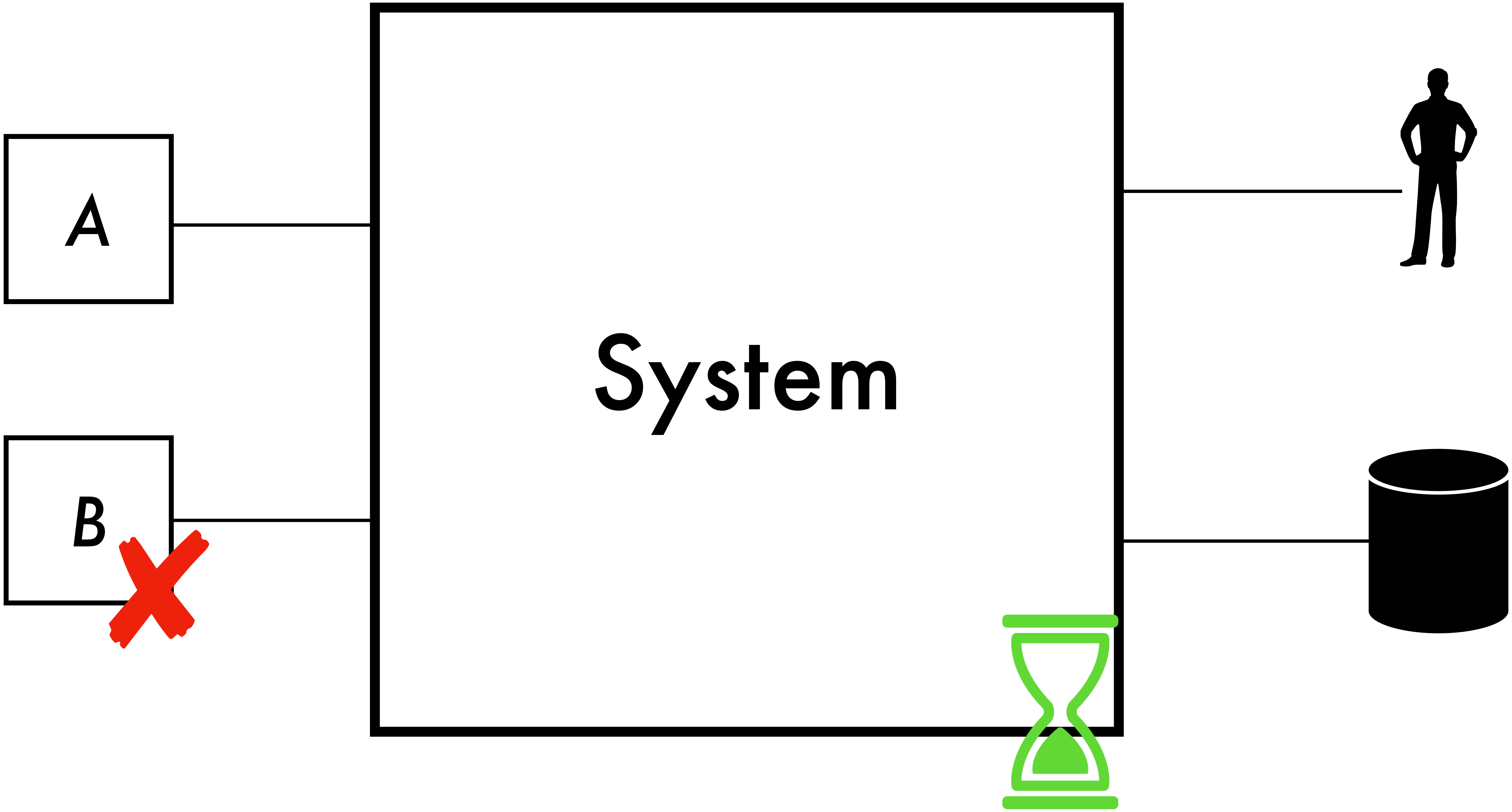
Crash



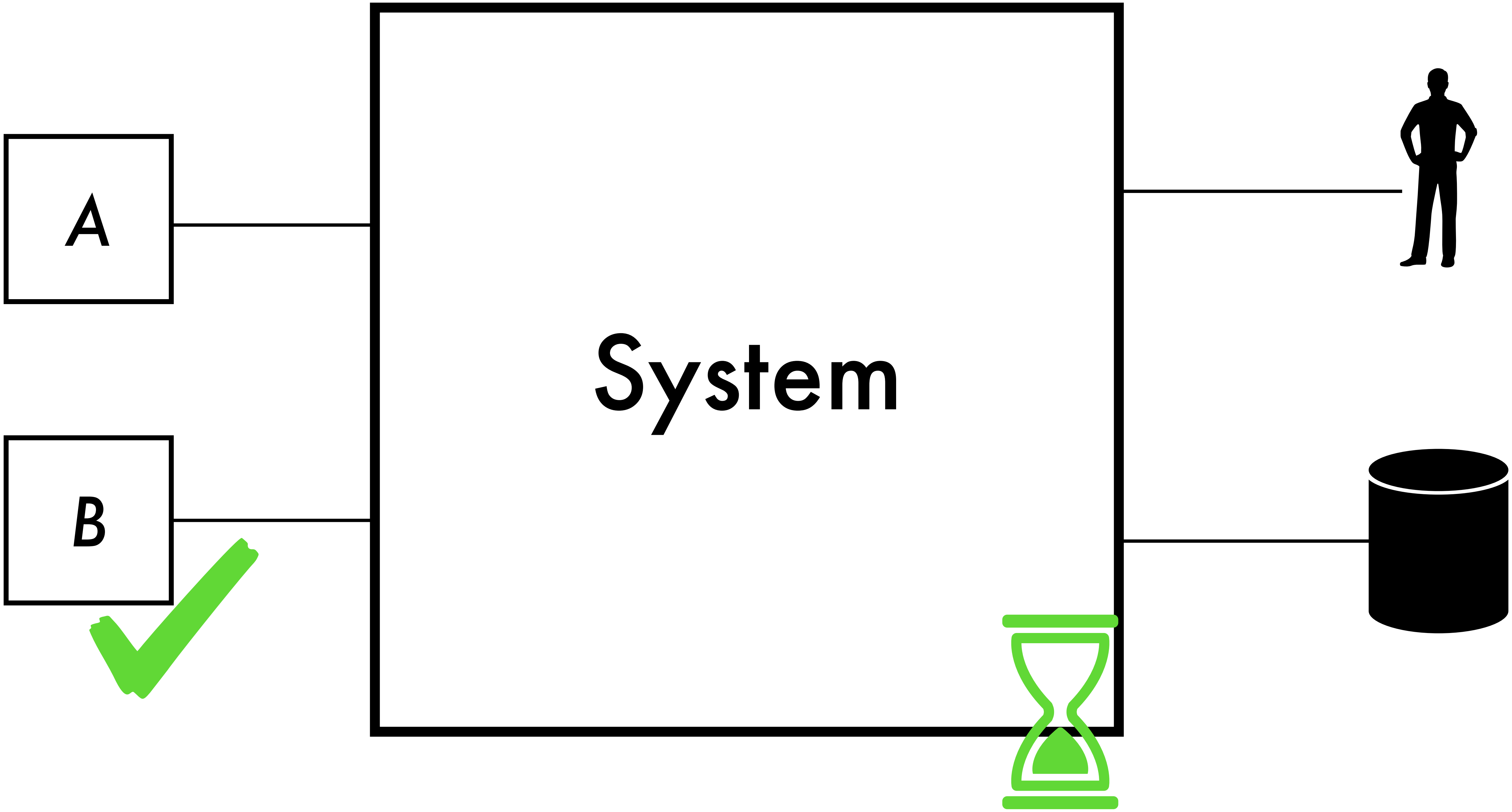
Restart



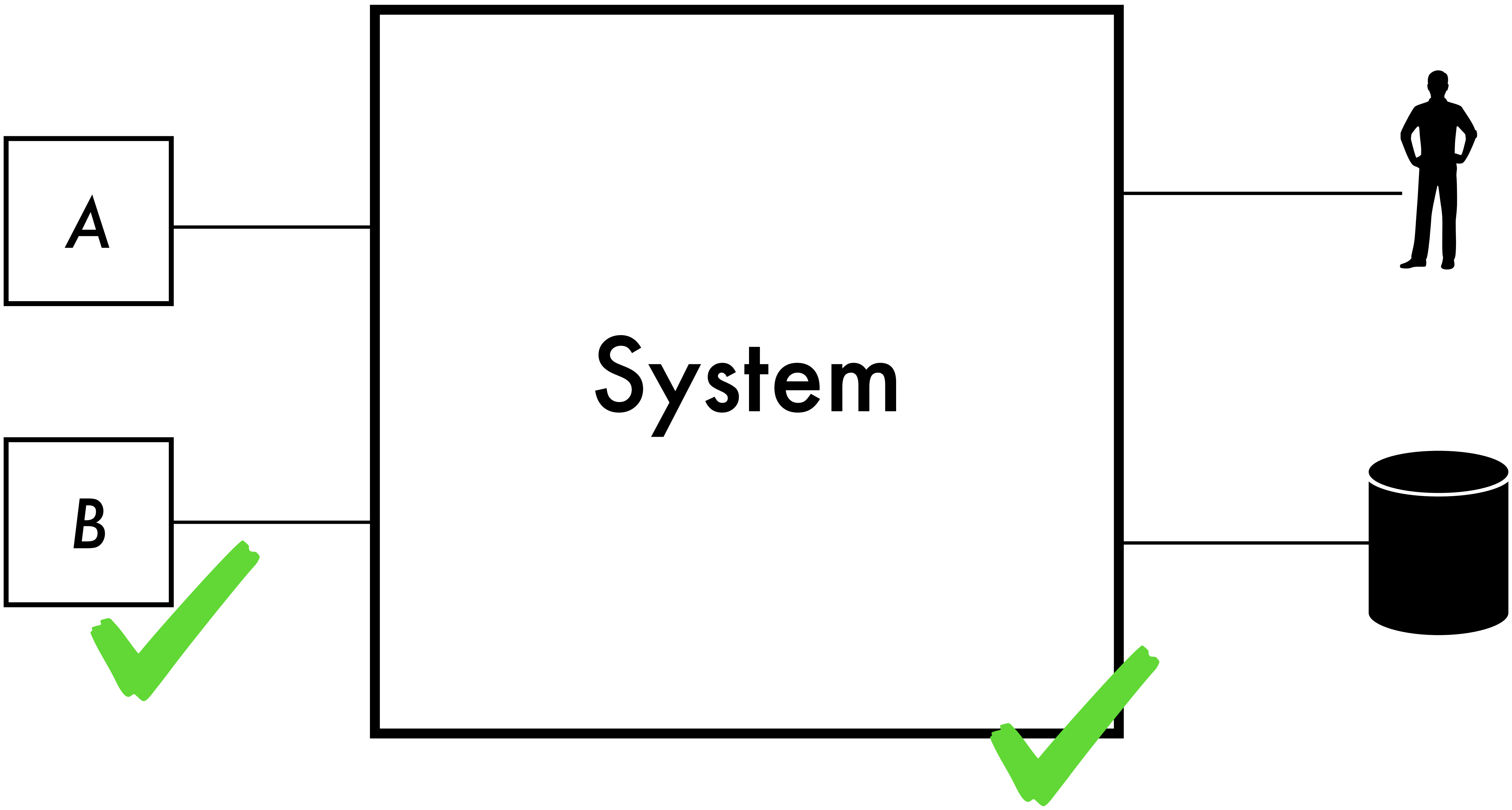
Wait



External up



Continue



A

B

System

“faulty documentation”

Computer ↔ **Radar system**

NASA

Grumman Aerospace

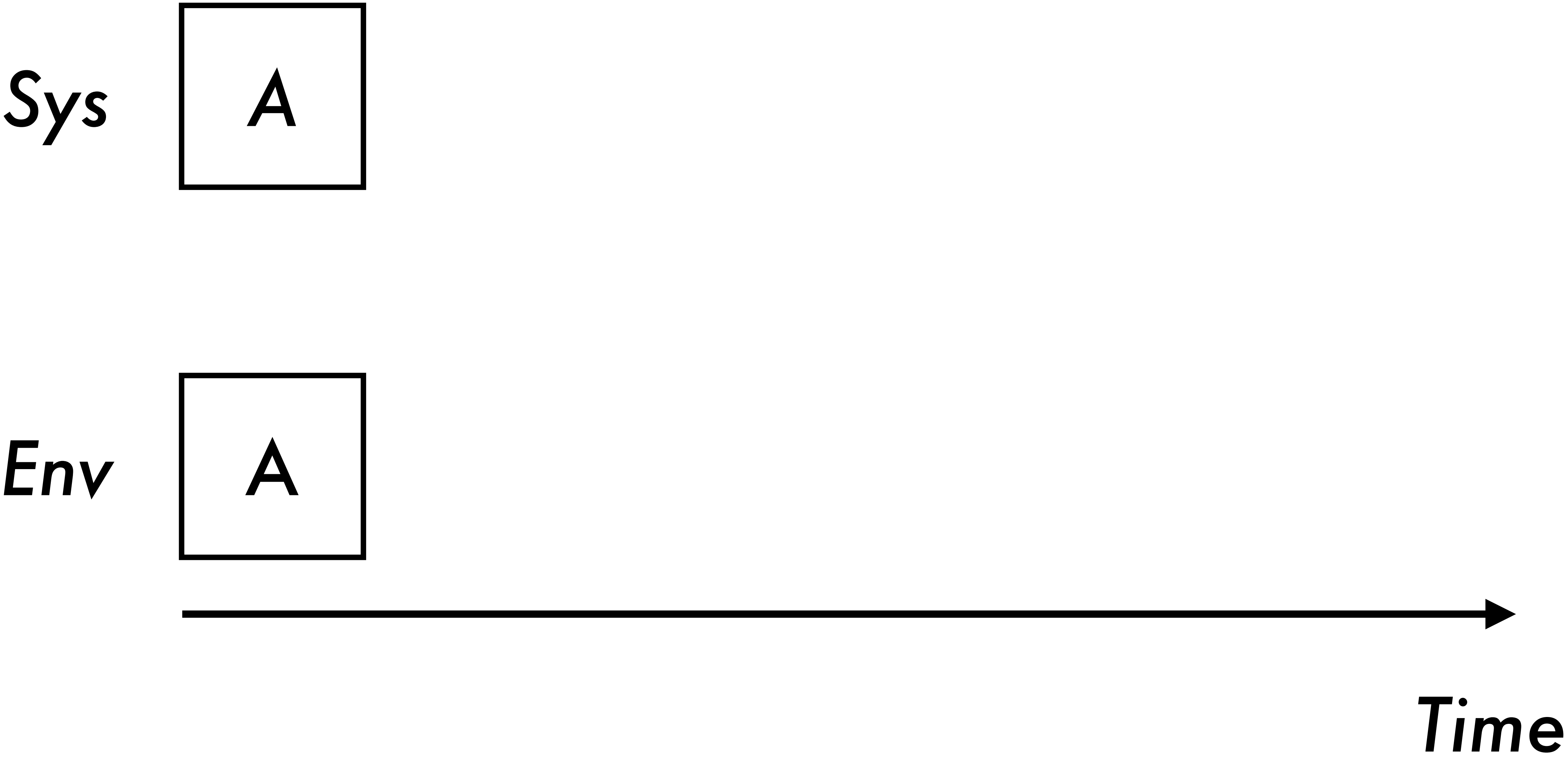
Interface Control Document:

☑ frequency locked

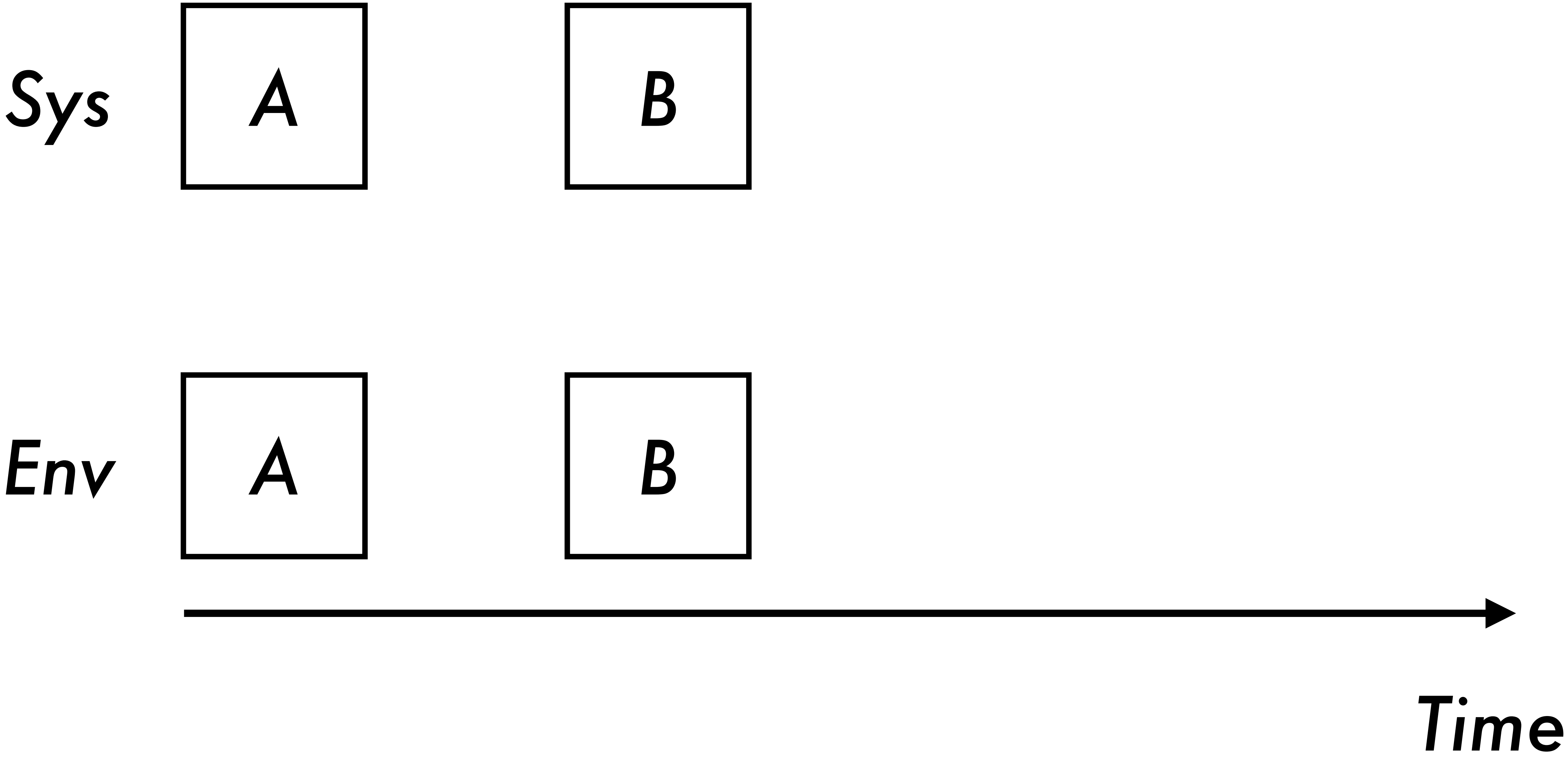
□ phase synchronized

Long running processes

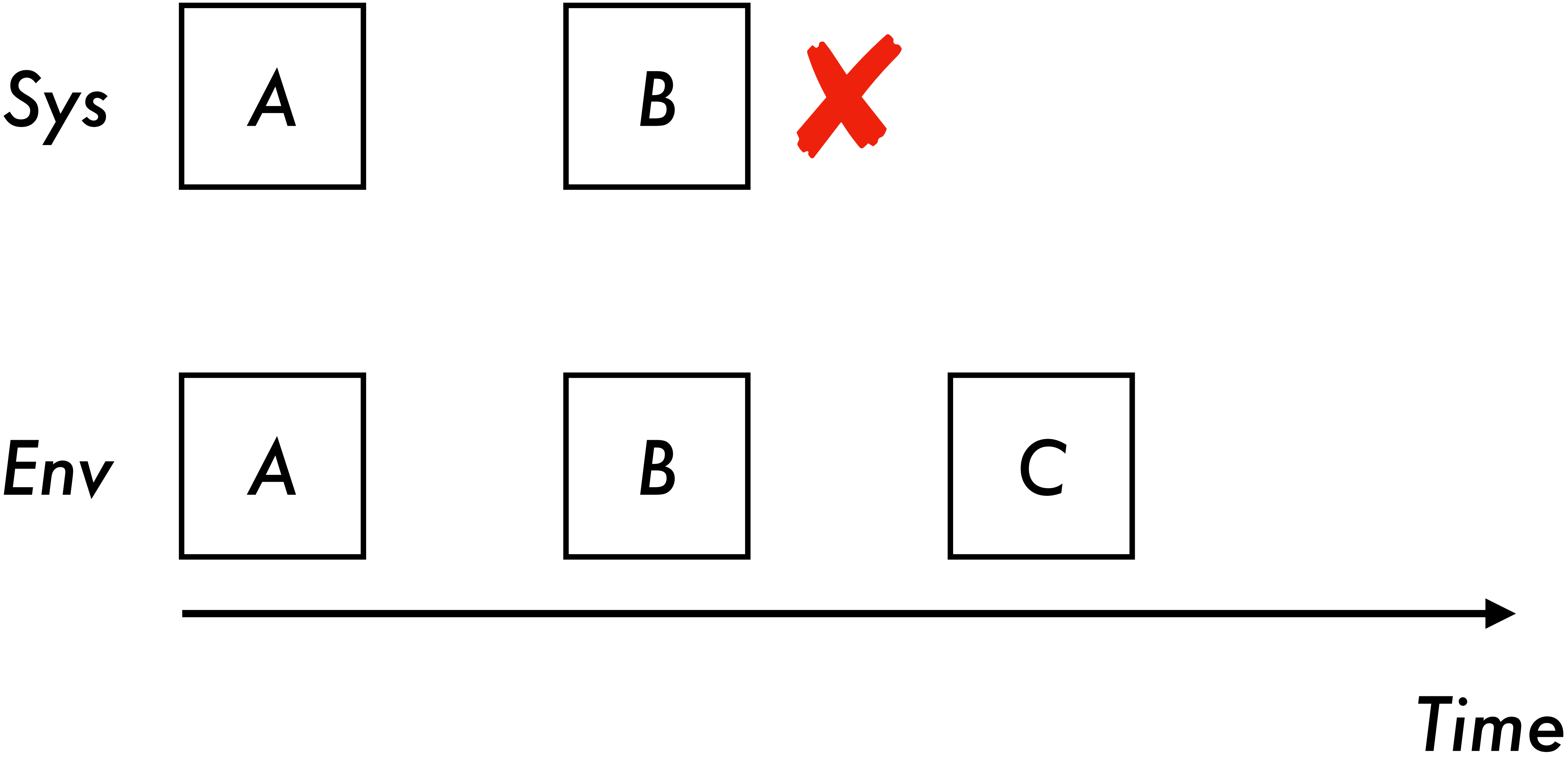
Long running process



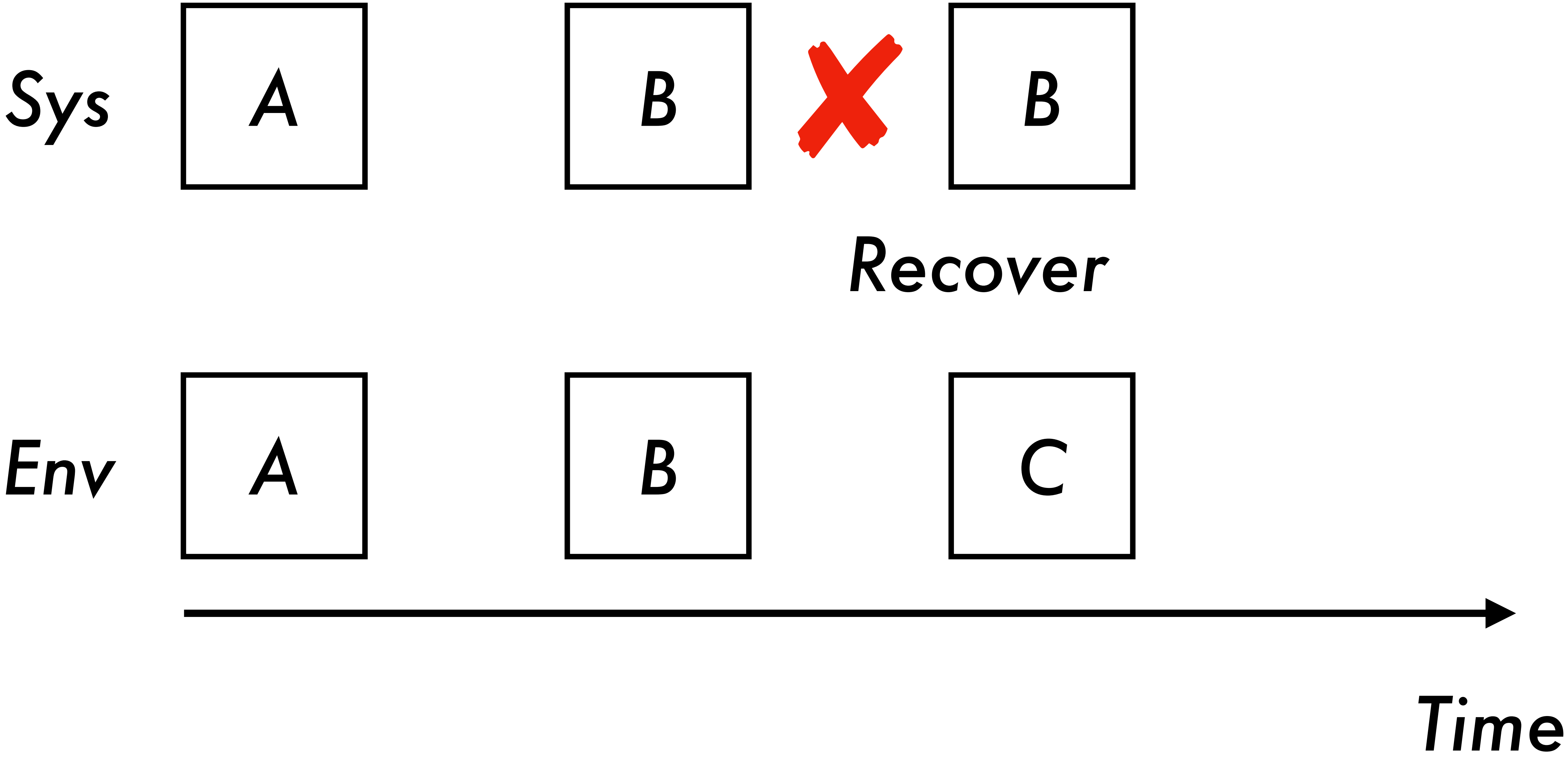
Long running process



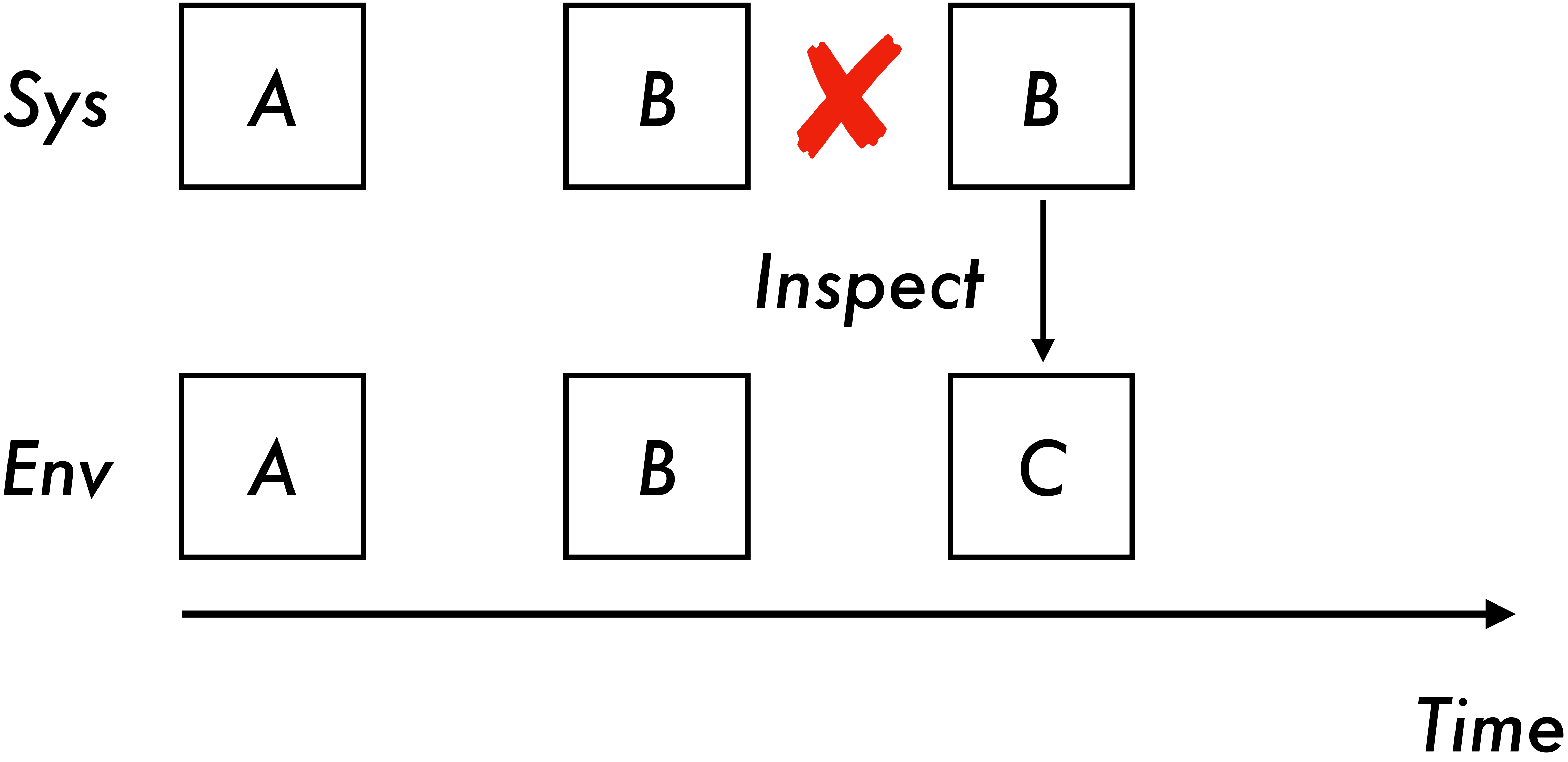
Long running process



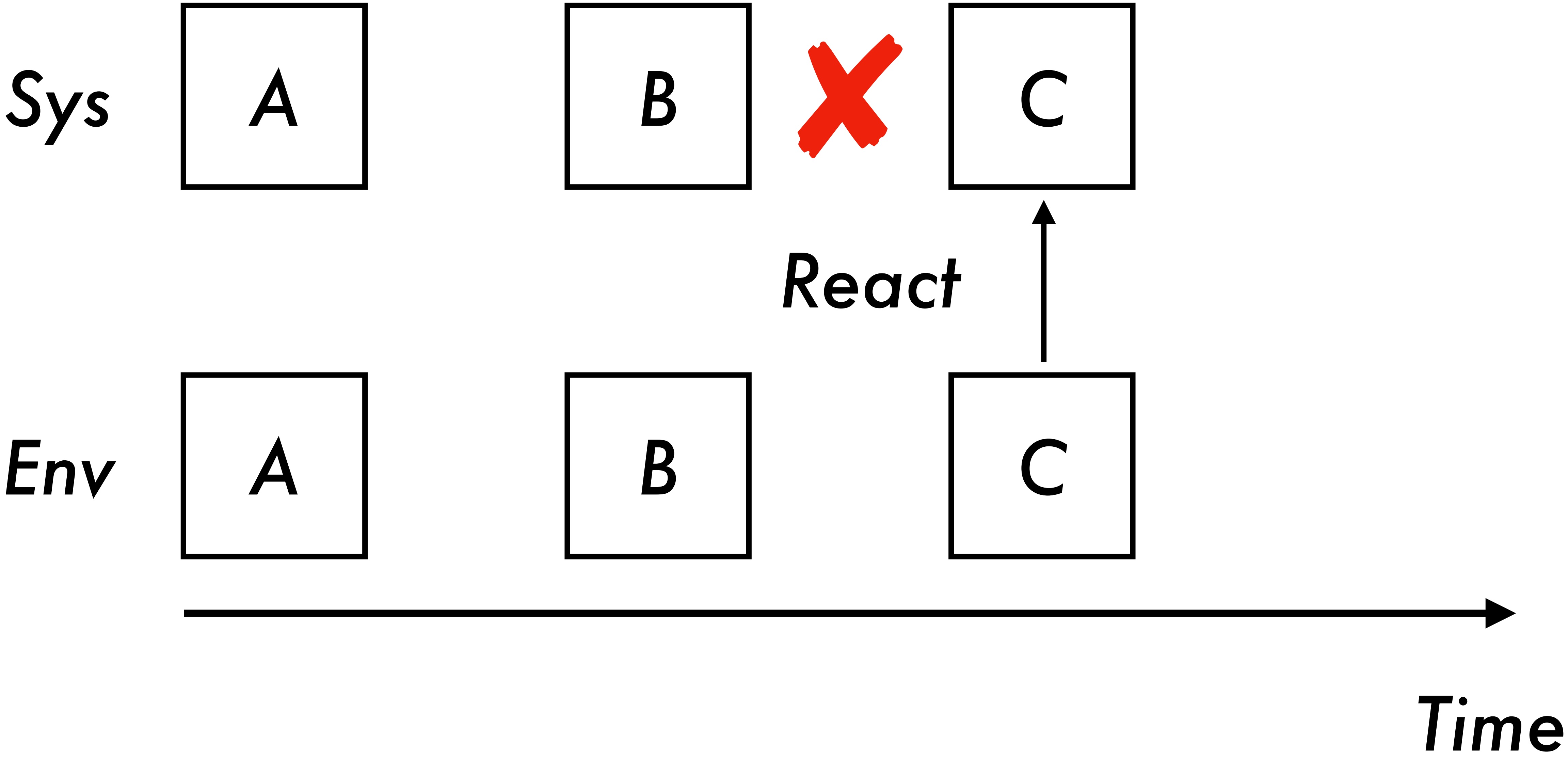
Long running process



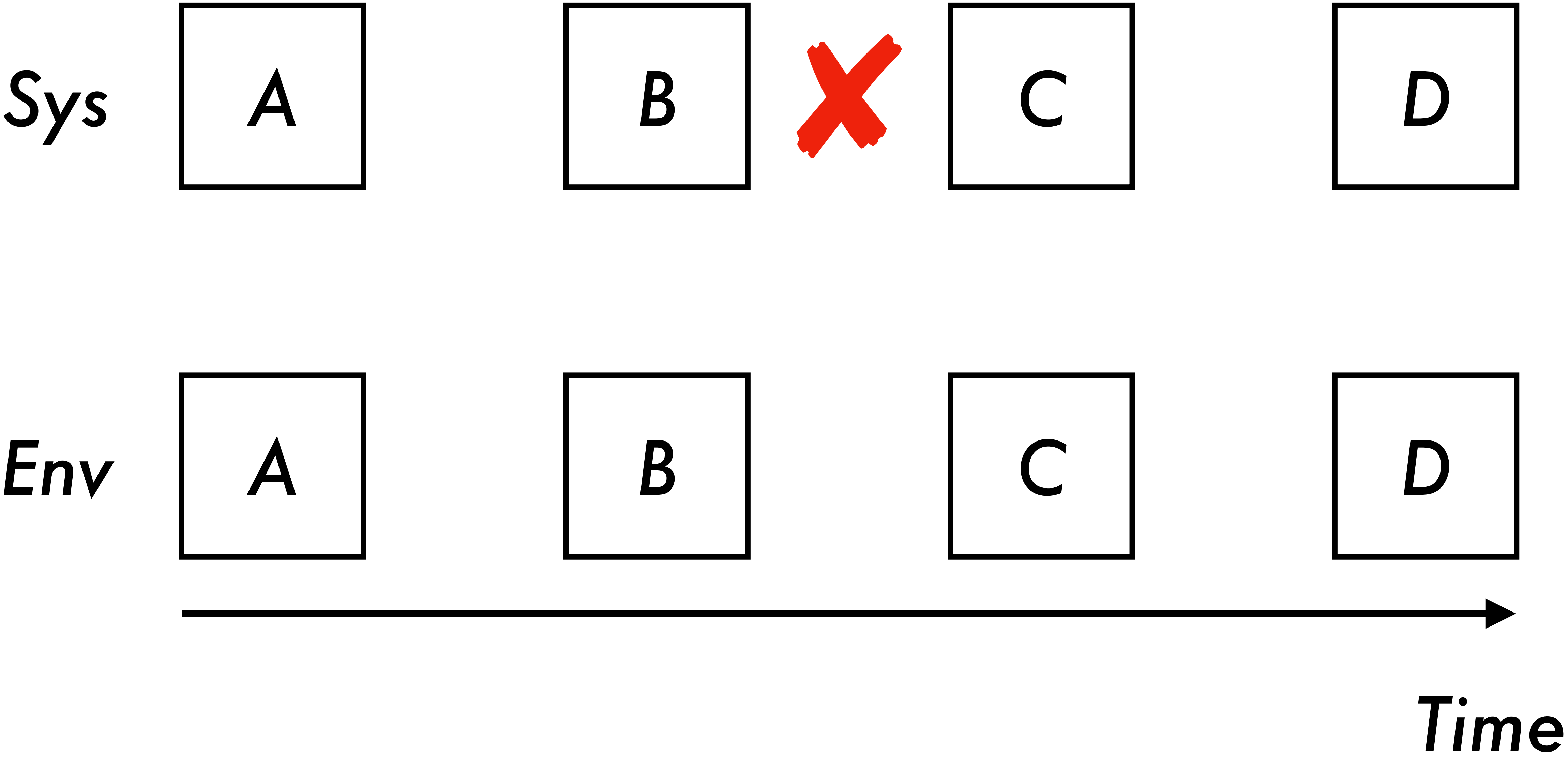
Long running process



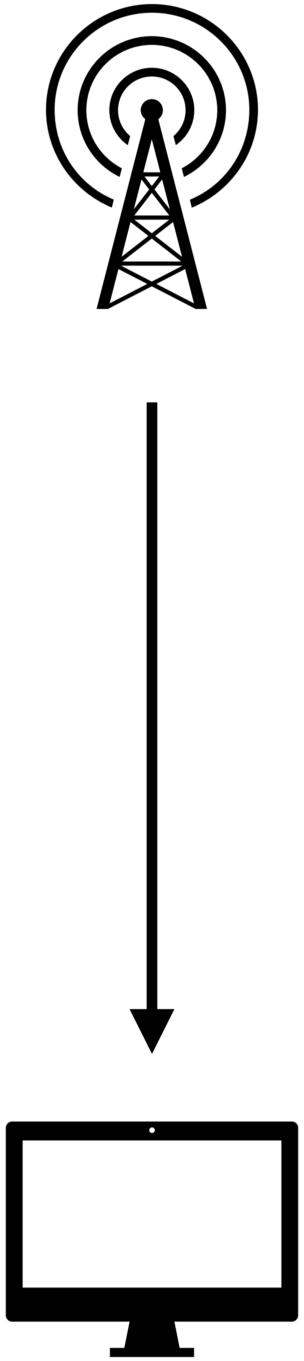
Long running process



Long running process



“With a 10% margin and a 13% drain, the LGC simply did not have enough CPU time to perform all the functions that were required.”

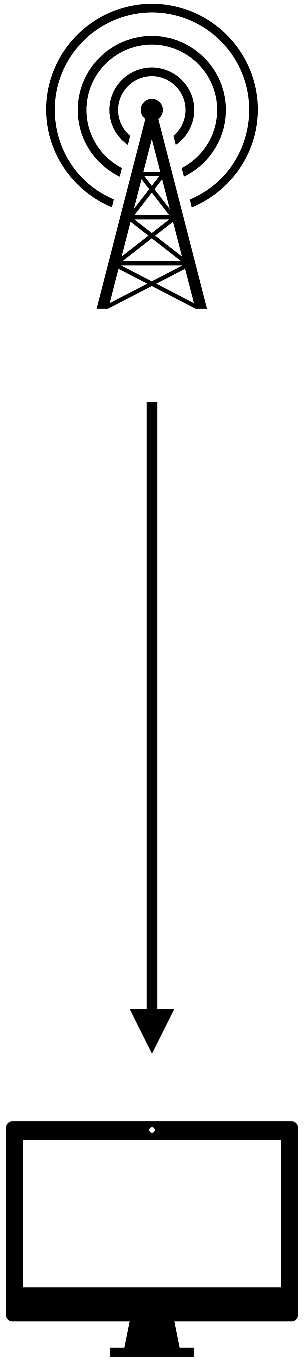


Radar System

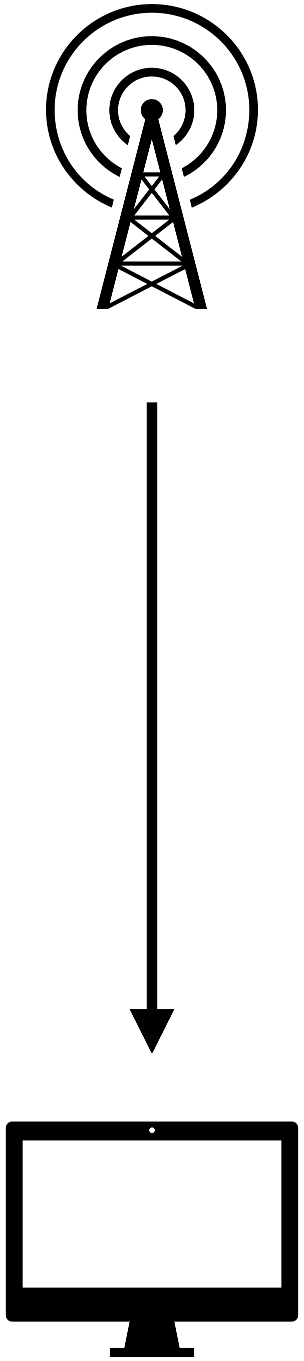
Computer

Radar System

90° or 270°



Computer

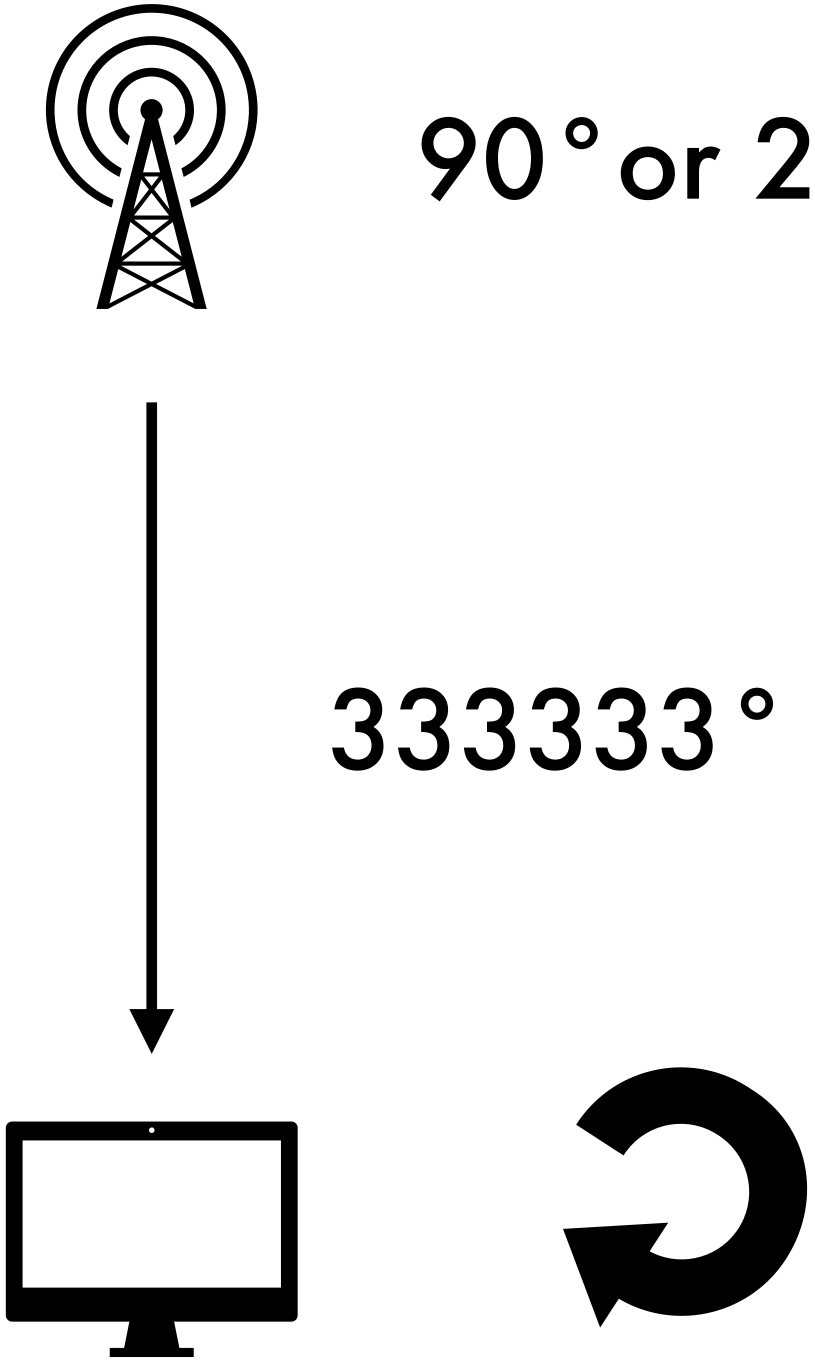


Radar System

90° or 270°

333333°

Computer



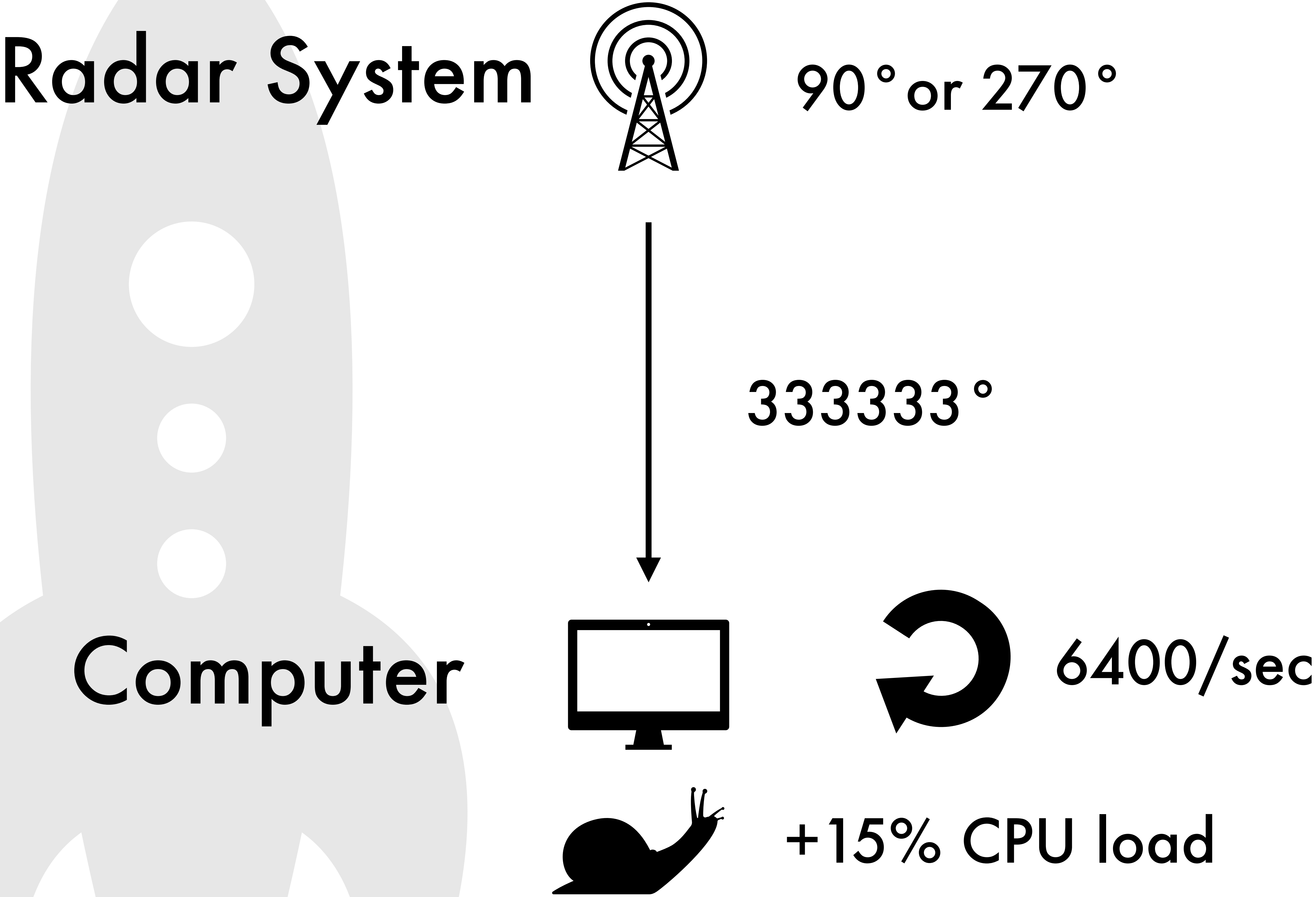
Radar System

90° or 270°

333333°

Computer

6400/sec



Radar System

90° or 270°

333333°

Computer

6400/sec

+15% CPU load

Questions?

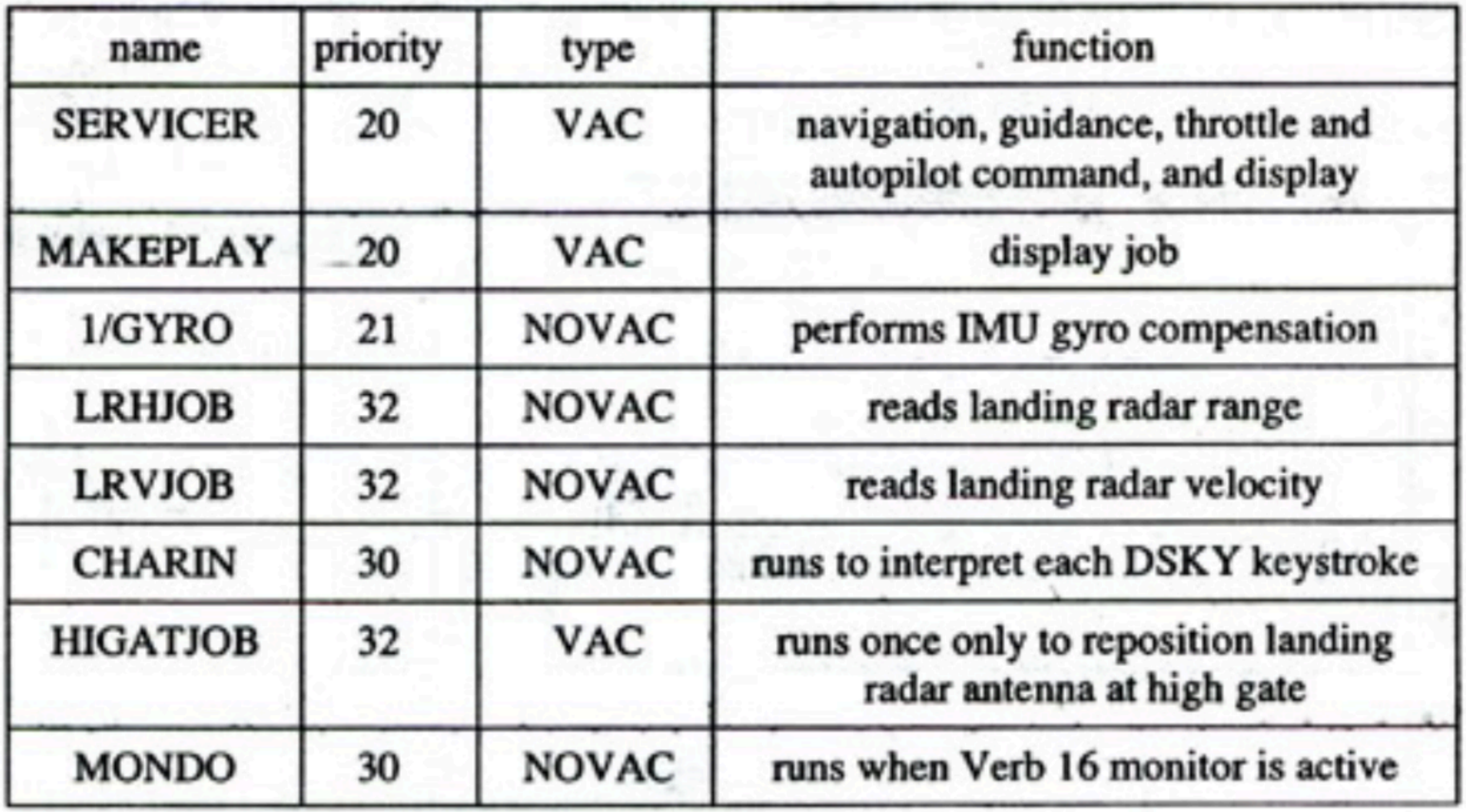
Conclusion: Apollo software

Core design principles

1. Prioritized jobs

1. Prioritized Jobs

Low



name	priority	type	function
SERVICER	20	VAC	navigation, guidance, throttle and autopilot command, and display
MAKEPLAY	20	VAC	display job
1/GYRO	21	NOVAC	performs IMU gyro compensation
LRHJOB	32	NOVAC	reads landing radar range
LRVJOB	32	NOVAC	reads landing radar velocity
CHARIN	30	NOVAC	runs to interpret each DSKY keystroke
HIGATJOB	32	VAC	runs once only to reposition landing radar antenna at high gate
MONDO	30	NOVAC	runs when Verb 16 monitor is active

High

Core design principles

1. Prioritized jobs
2. Restart protection

2. Restart Protection

```
NEW_X = X + 1
restister waypoint
X = NEW_X
```

Core design principles

1. Prioritized jobs
2. Restart protection
3. Crash if resources exceeded

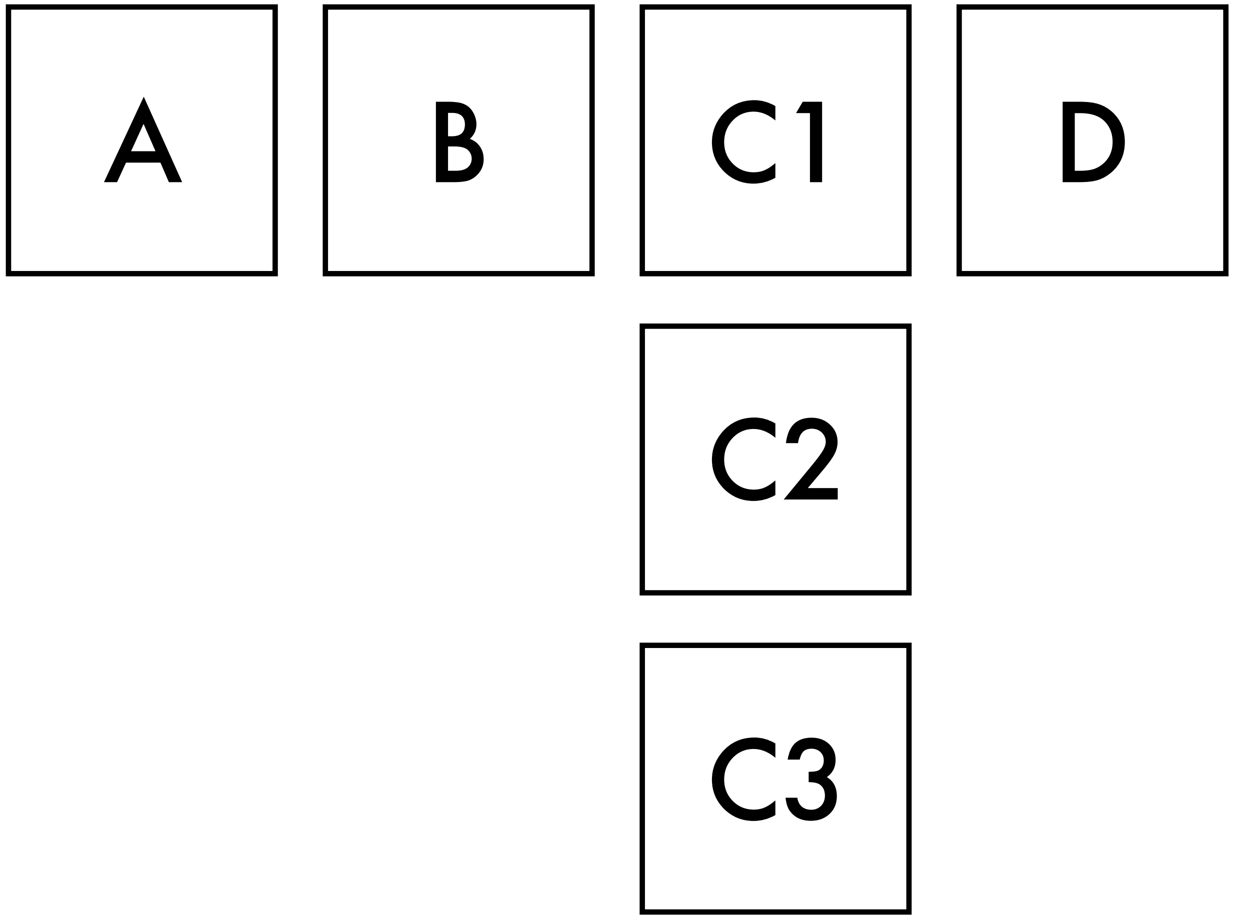
3. Crash if resources exceeded

"When the next request was made the Executive, unable to comply, called BAILOUT with a 1201 or 1202 alarm code."

Core design principles

1. Prioritized jobs
2. Restart protection
3. Crash if resources exceeded
4. Restore only most recent copy of a job

4. Restore only most recent copy of a job

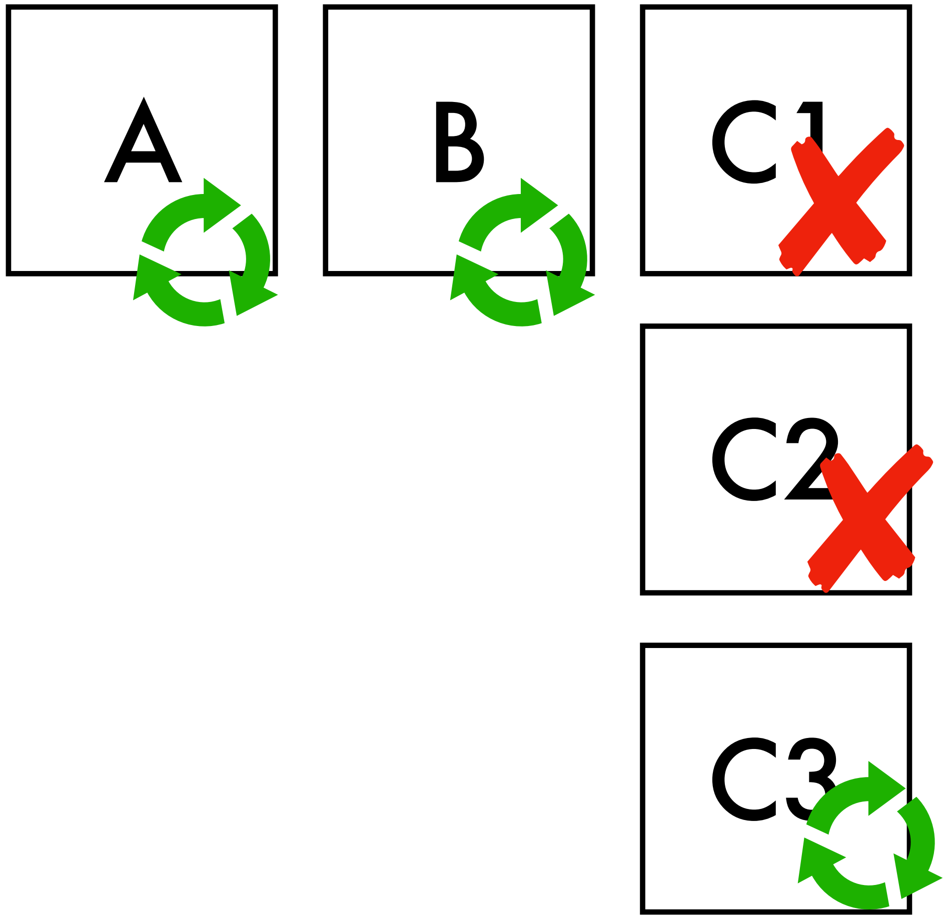


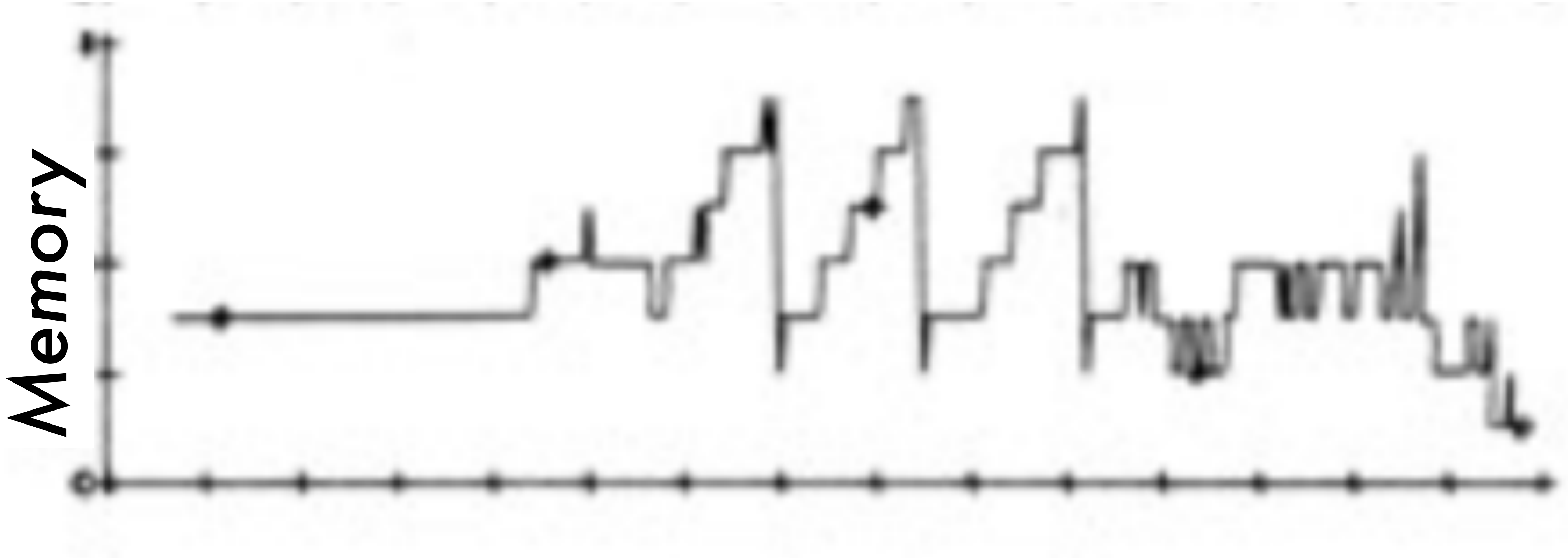
A B C1 D

C2

C3

4. Restore only most recent copy of a job





"The interesting effect of this train of events... was that the problem fixed itself."

References

1. Tales from the Lunar Module Guidance Computer, Don Eyles, 2004. Link.

2. The Charming Genius of the Apollo Guidance Computer, Video by Brian Troutwine. Link.

3. Why Do Computers Stop and What Can Be Done About It? JimGray, 1985. PDF. Link.

4. Crash-Only Software. George Candea & Armando Fox, 2003. PDF. Link.

5. Long Running Processes with Event Sourcing and CQRS. Andriy Drozdyuk 2016. Link.

"...at MET 102:45:40, Armstrong landed the spacecraft safely in the Sea of Tranquility."

The end

Contact

@andriyko on Twitter