dependence & dependability

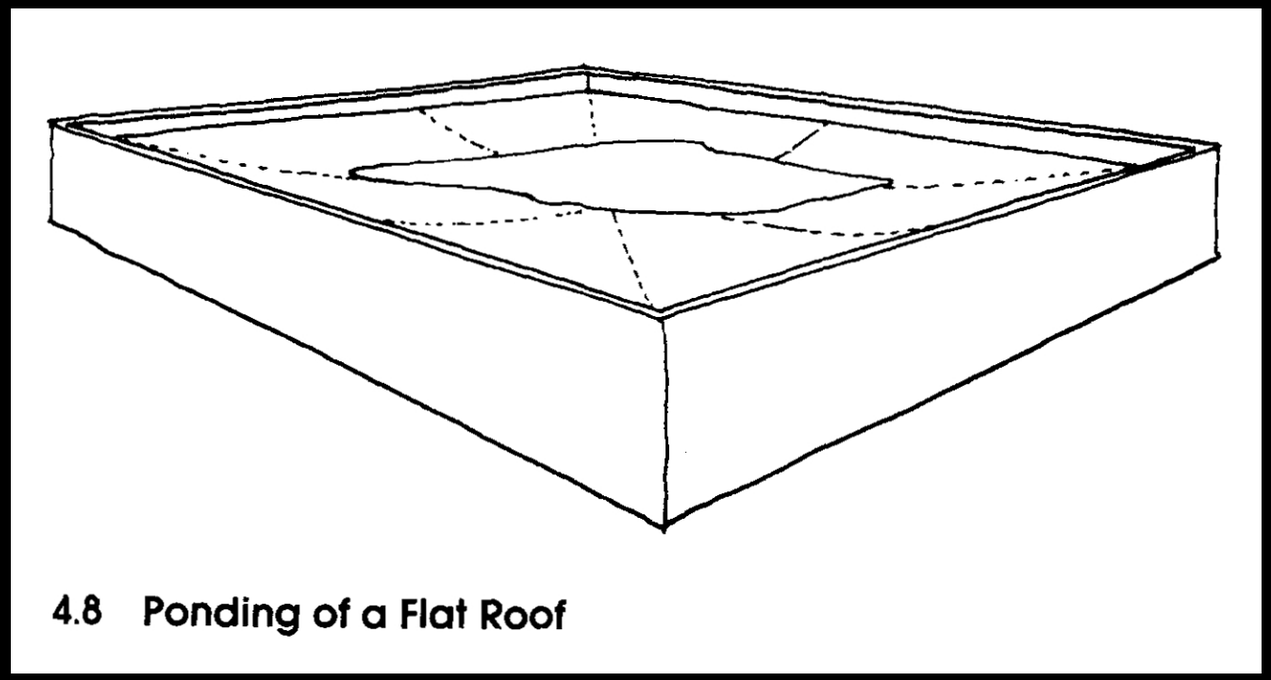
IBM Research · June 1, 2009

Daniel Jackson & Eunsuk Kang, MIT CSAIL

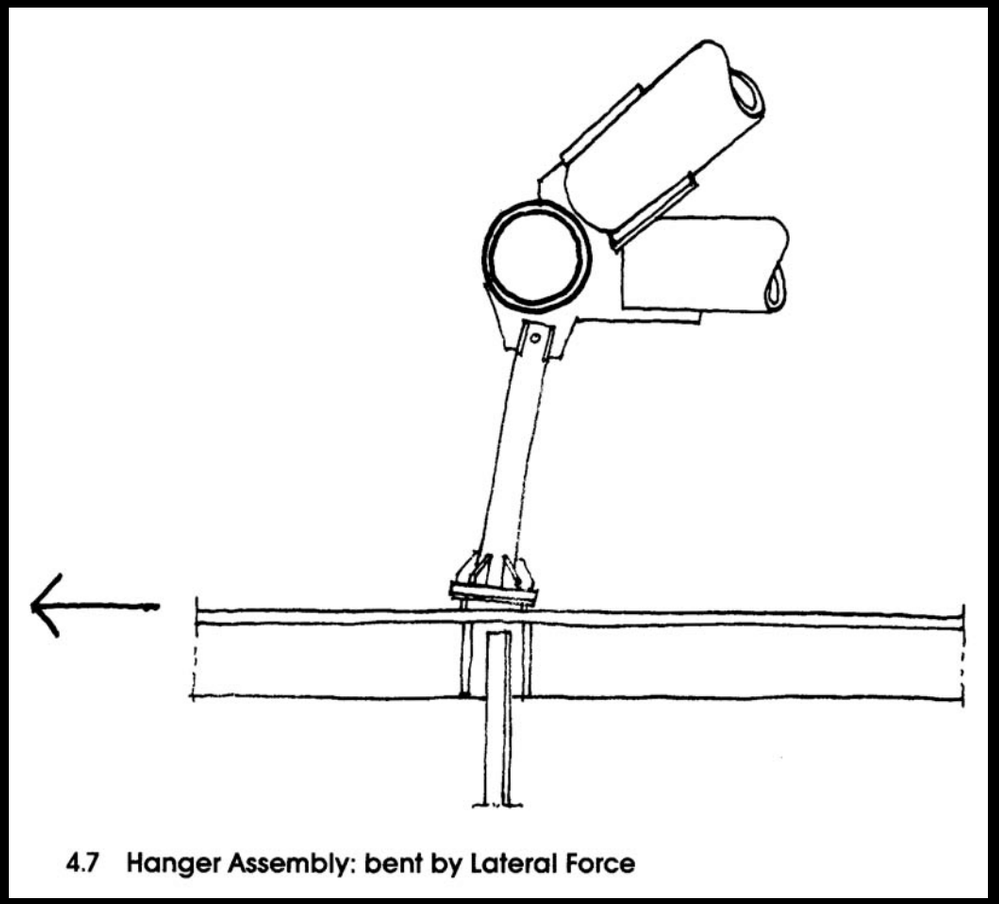
kemper arena, kansas city, 2007

kemper arena, 1979

what happened?



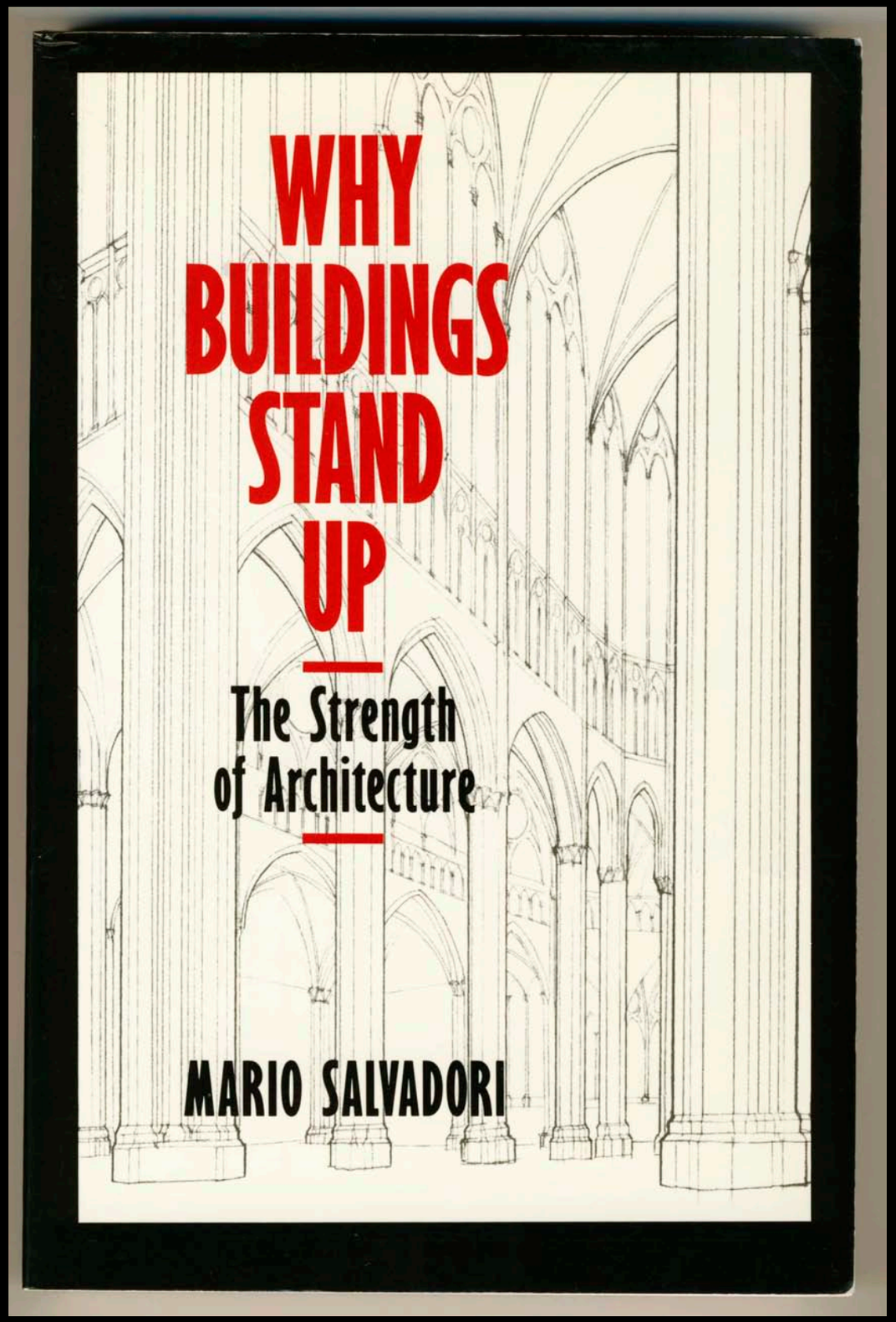
4.8 Ponding of a Flat Roof



4.7 Hanger Assembly: bent by Lateral Force

Levy & Salvadori, *Why Buildings Fall Down*

failure = flawed success story



Therac 25

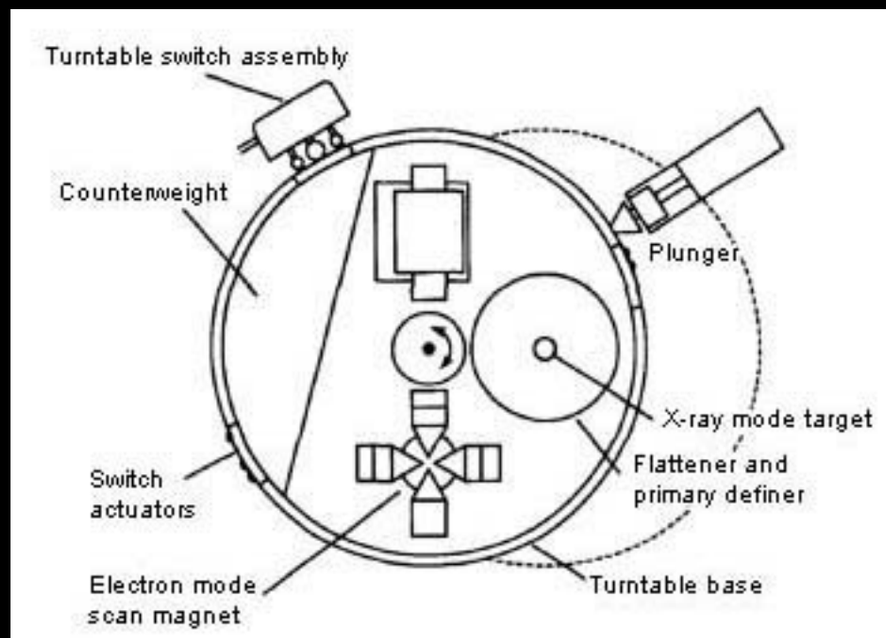


Figure B. Upper turntable assembly

AECL fault tree analysis (1983)

did not include software

P(computer selects wrong energy) = 10^{-11}

Leveson & Turner (1993)

race conditions, lack of interlocks, etc

A Direct Path to Dependable Software, CACM, March 2009

wordle thanks to Jonathan Feinberg, IBM Research, Cambridge

goals

a notation for
analyzing, justifying, explaining

desiderata
simple, intuitive, graphical
support formal analysis

the uses relation

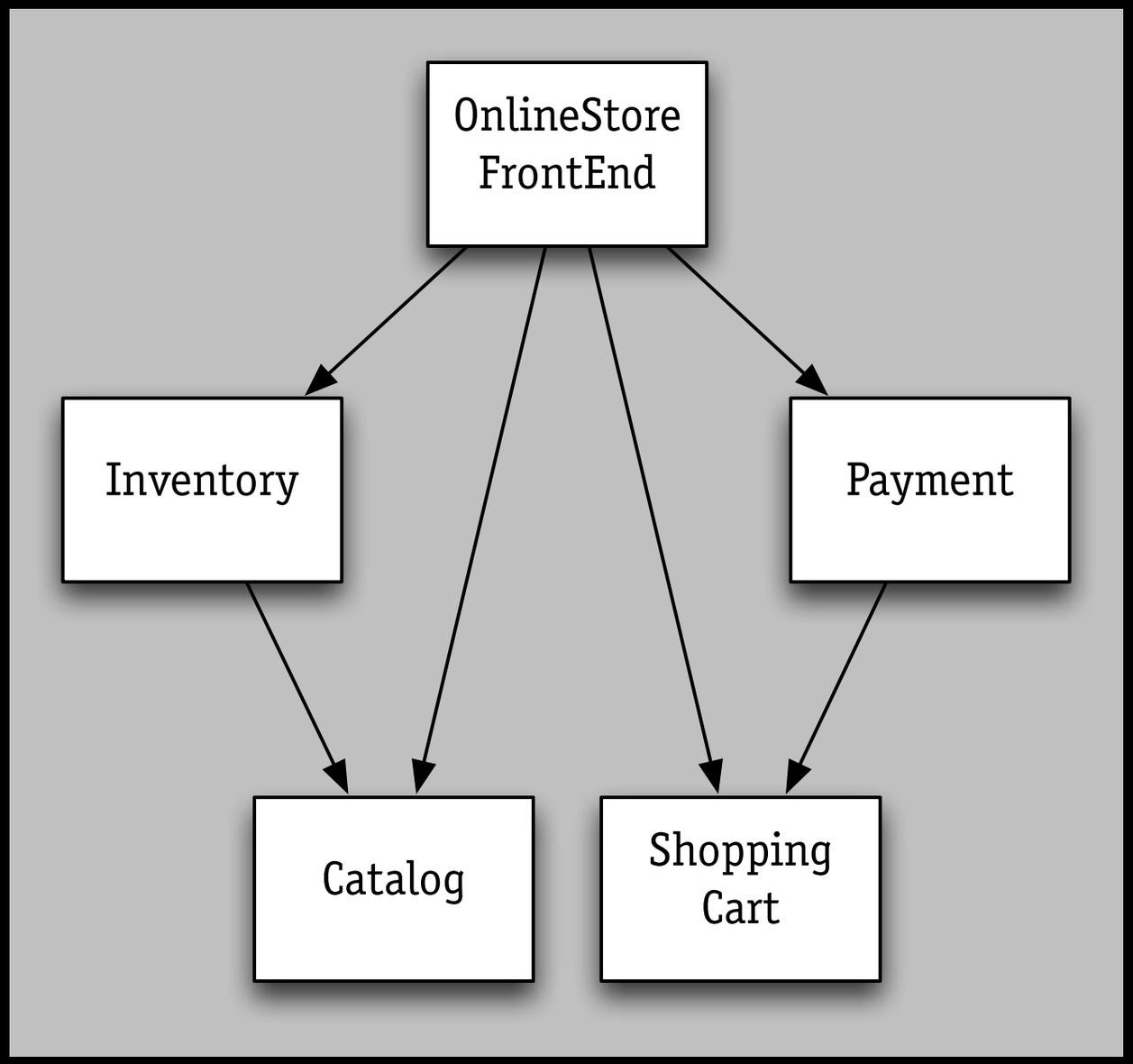
the uses relation

A *uses* B if there exist situations in which the correct functioning of A depends on the availability of a correct implementation of B

David Parnas

Designing Software for Ease of Extension and Contraction, 1979

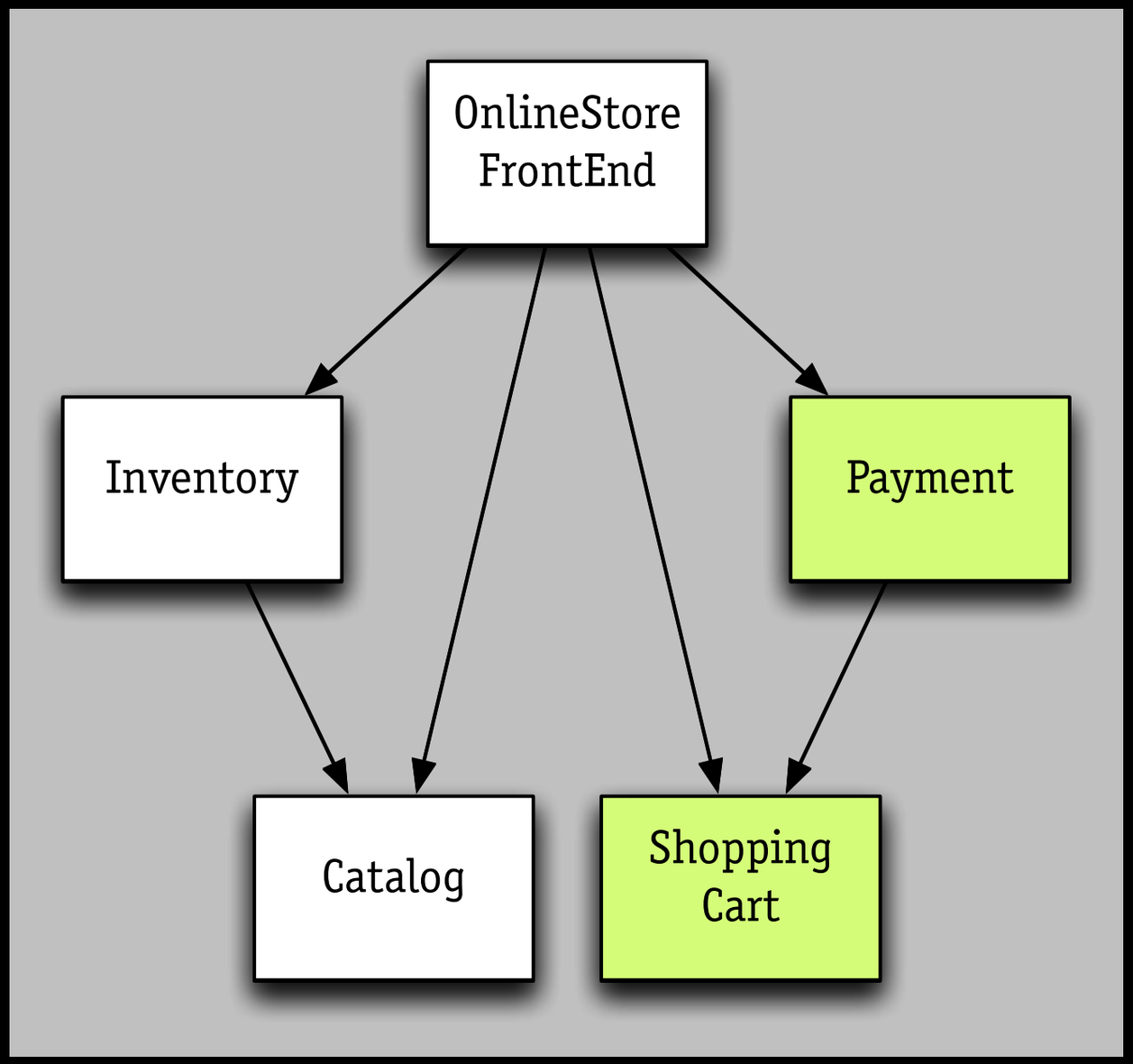
using uses



modular reasoning

› correct function of Payment depends only on ShoppingCart

using uses



modular reasoning

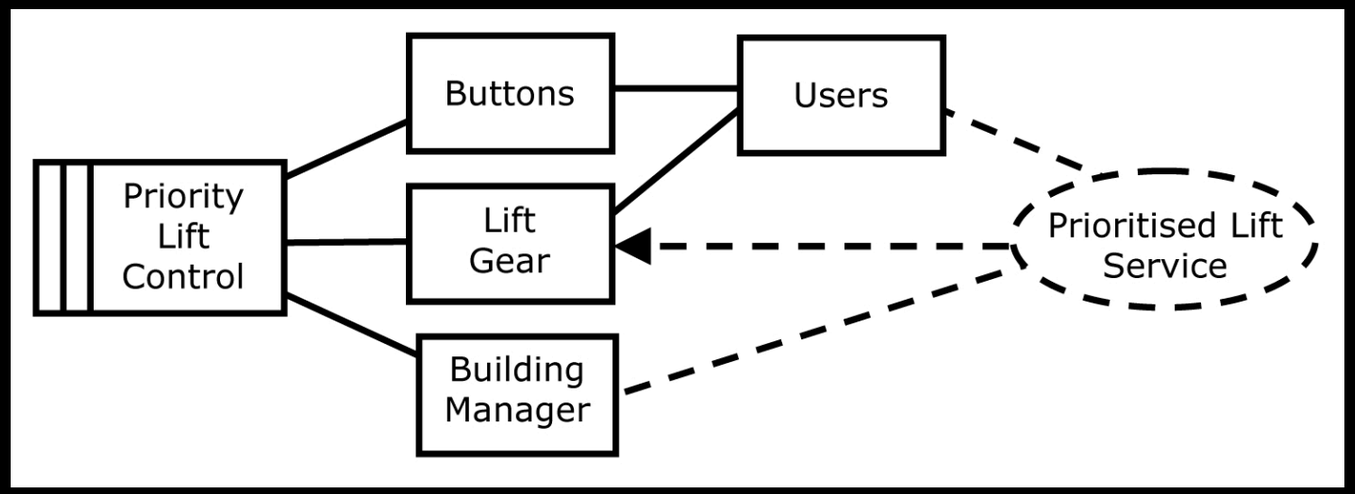
› correct function of Payment depends only on ShoppingCart

the notation

There probably isn't a best way to build the system, or even any major part of it; much more important is to avoid choosing a terrible way, and to have a clear division of responsibilities among the parts.

Butler Lampson
Hints for computer system design (1983)

key idea



inspired by

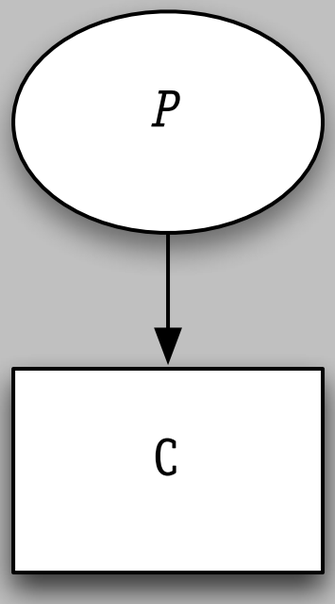
problem diagrams [Michael Jackson]

represent

properties, components, their relationship

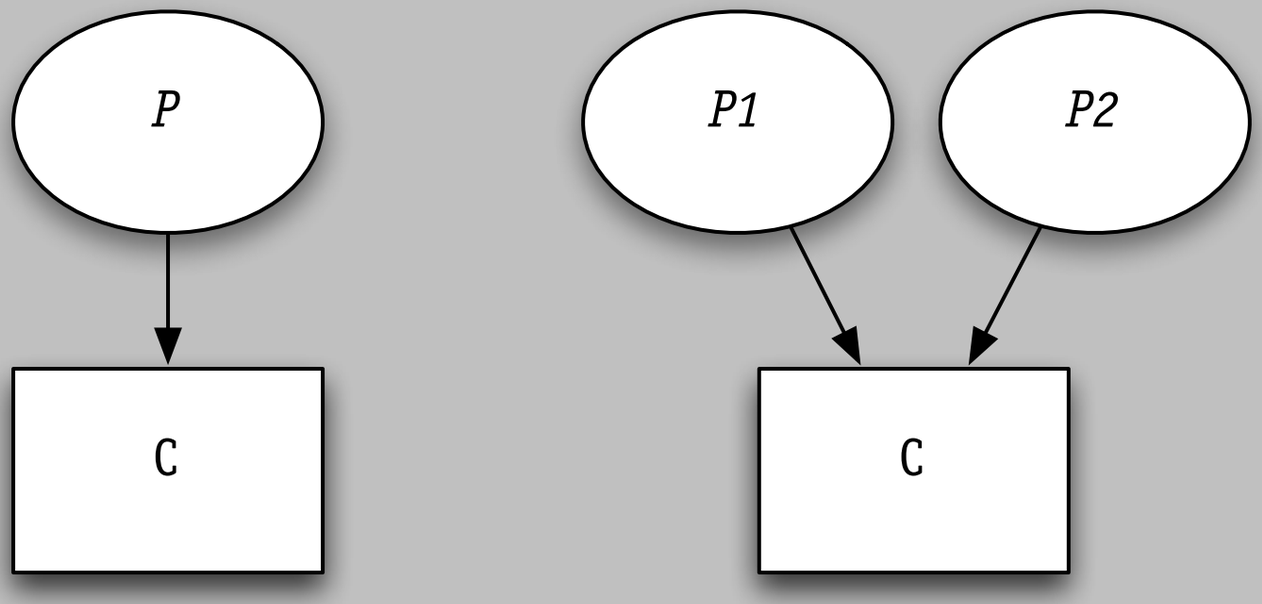
properties & components

properties & components



a specification
is a property

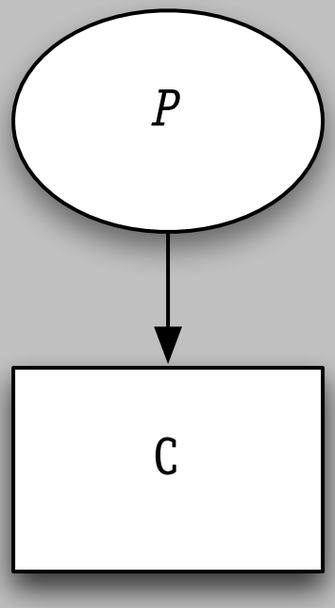
properties & components



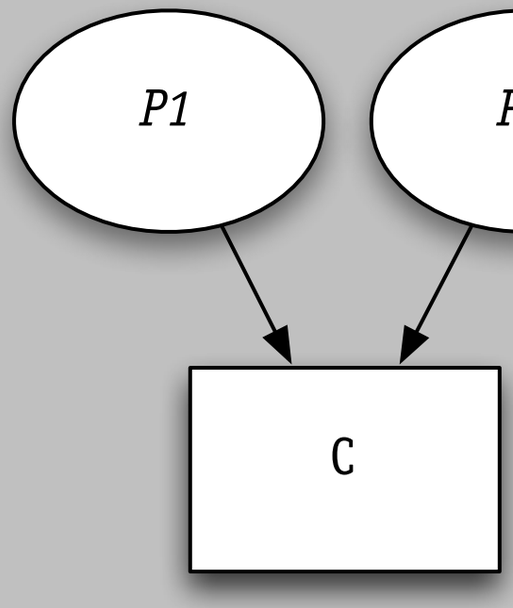
a specification
is a property

a component may
satisfy >1 property

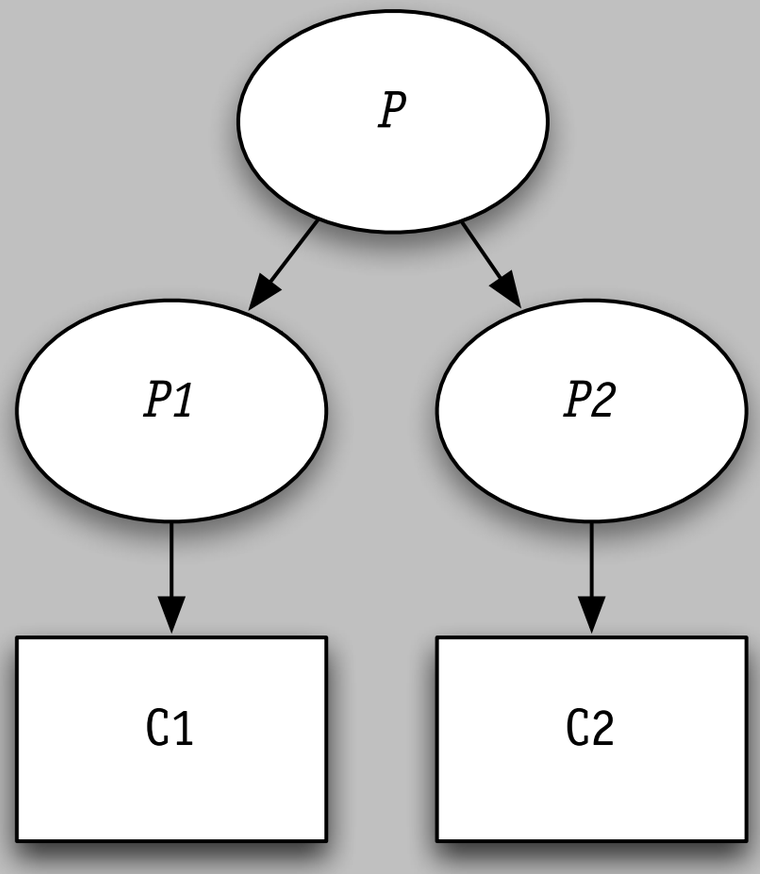
properties & components



a specification
is a property



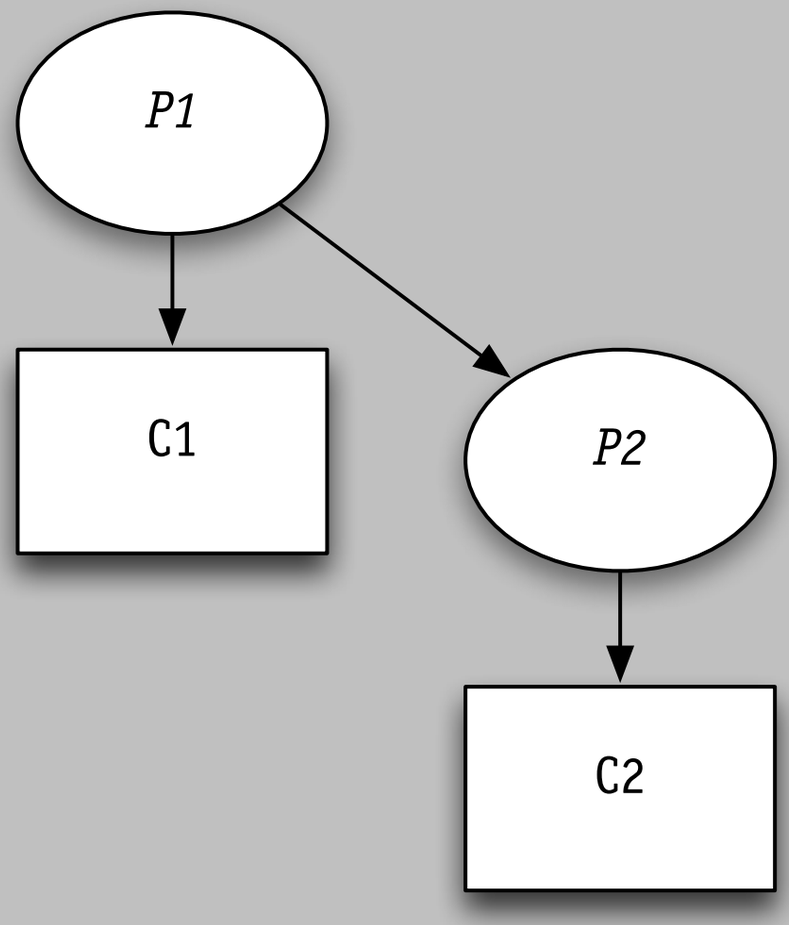
a component may
satisfy >1 property



components can be
justified independently
but achieve a common goal

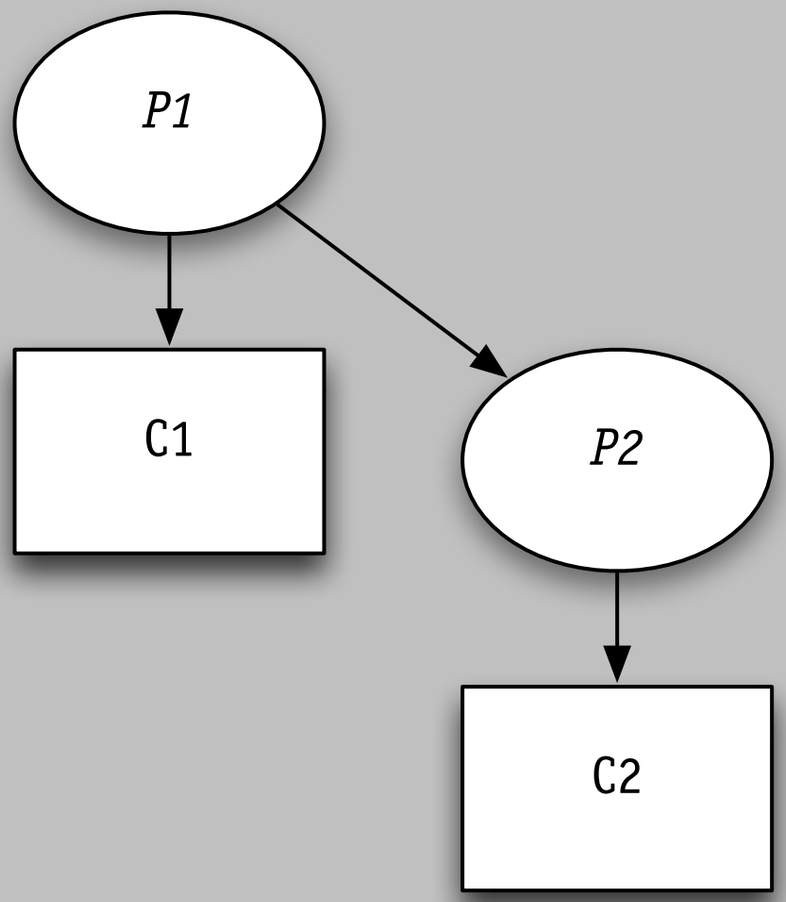
properties & components

properties & components

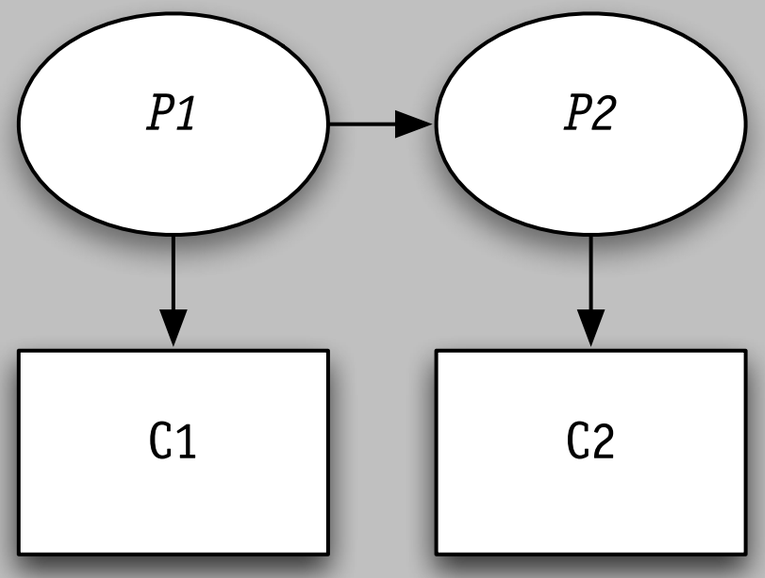


property established by component and property of another component

properties & components



property established by component and property of another component



equivalent diagram, less familiar layout

an example: tracking stocks

problem

track stocks with given set of ticker symbols and display message when move exceeds bound

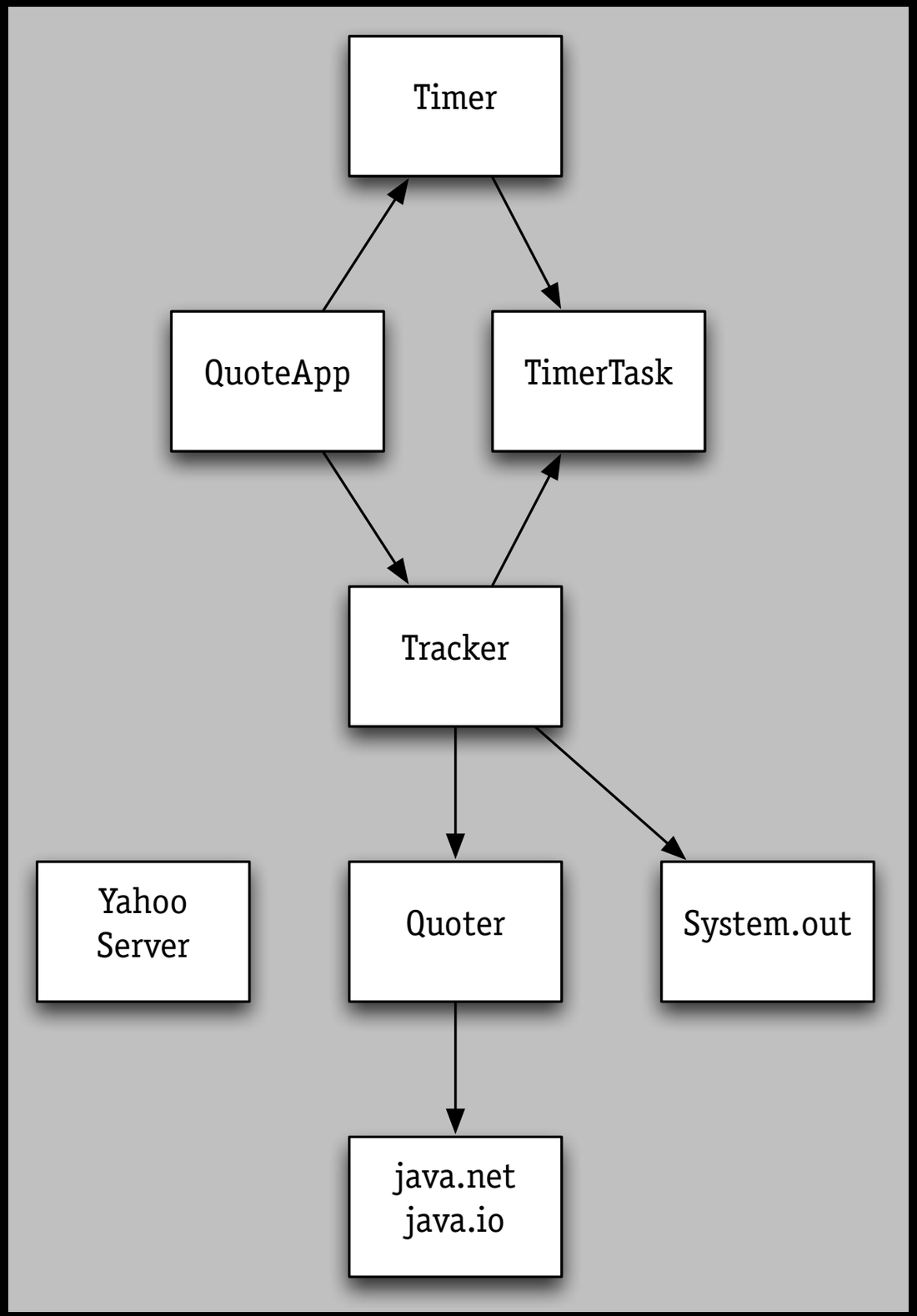
```
AAPL: now 12295 prev hi: 12295, prev lo: 12289
IBM: now 10218 prev hi: 10218, prev lo: 10212
INTC: now 1550 prev hi: 1552, prev lo: 1550
```

```
public class QuoteApp {
    public static void main(String[] args) throws Exception {
     Timer timer =    new Timer();
        for (String ticker: args)
         timer.schedule (   new Tracker (ticker), 0, 10000);
    }
}
```

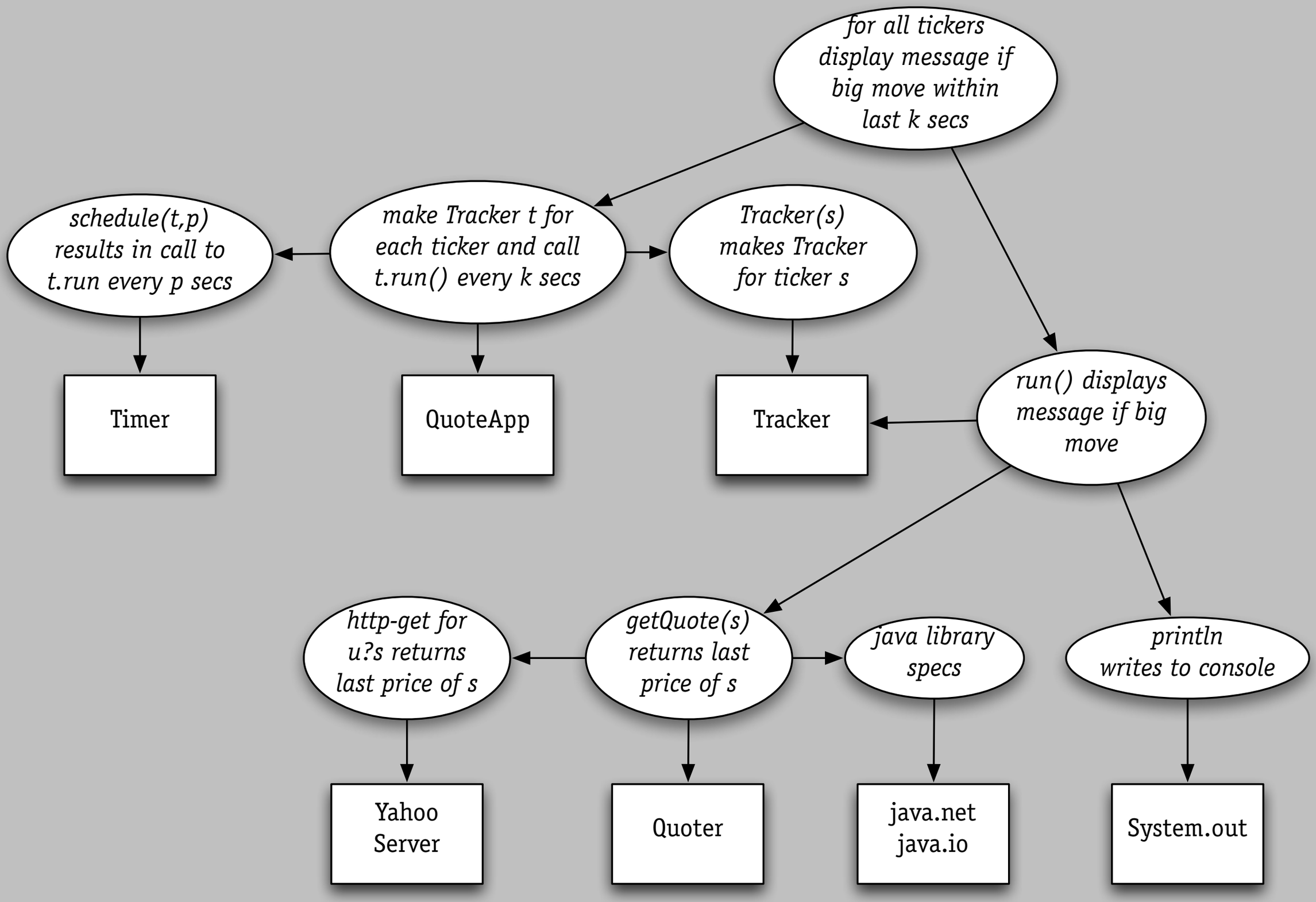
```
public class Tracker extends TimerTask {
 String    ticker;
    int hi = 0; int lo = Integer.MAX_VALUE;
    int MOVE = 1;

    public Tracker (String t) {ticker = t;}
    public void run () {
        int q = Quoter.getQuote(ticker);
        hi = Math.max(hi, q);
        lo = Math.min(lo, q);
        if (hi - lo > MOVE) {
         System.   out.println (ticker + ": now " + q + " prev hi: " + hi + ", prev lo: " + lo);
            hi = lo = q;
        }
    }
}
```

```
public class Quoter {
    public static int getQuote (String ticker) {
         URL url  =    new URL("http://finance.yahoo.com/d/quotes.csv?s=" + ticker + "&f=l1");
         String p =    new BufferedReader(new InputStreamReader(url.openStream())).readLine();
            return (int) (Float.valueOf (p) * 100);
    }
}
```

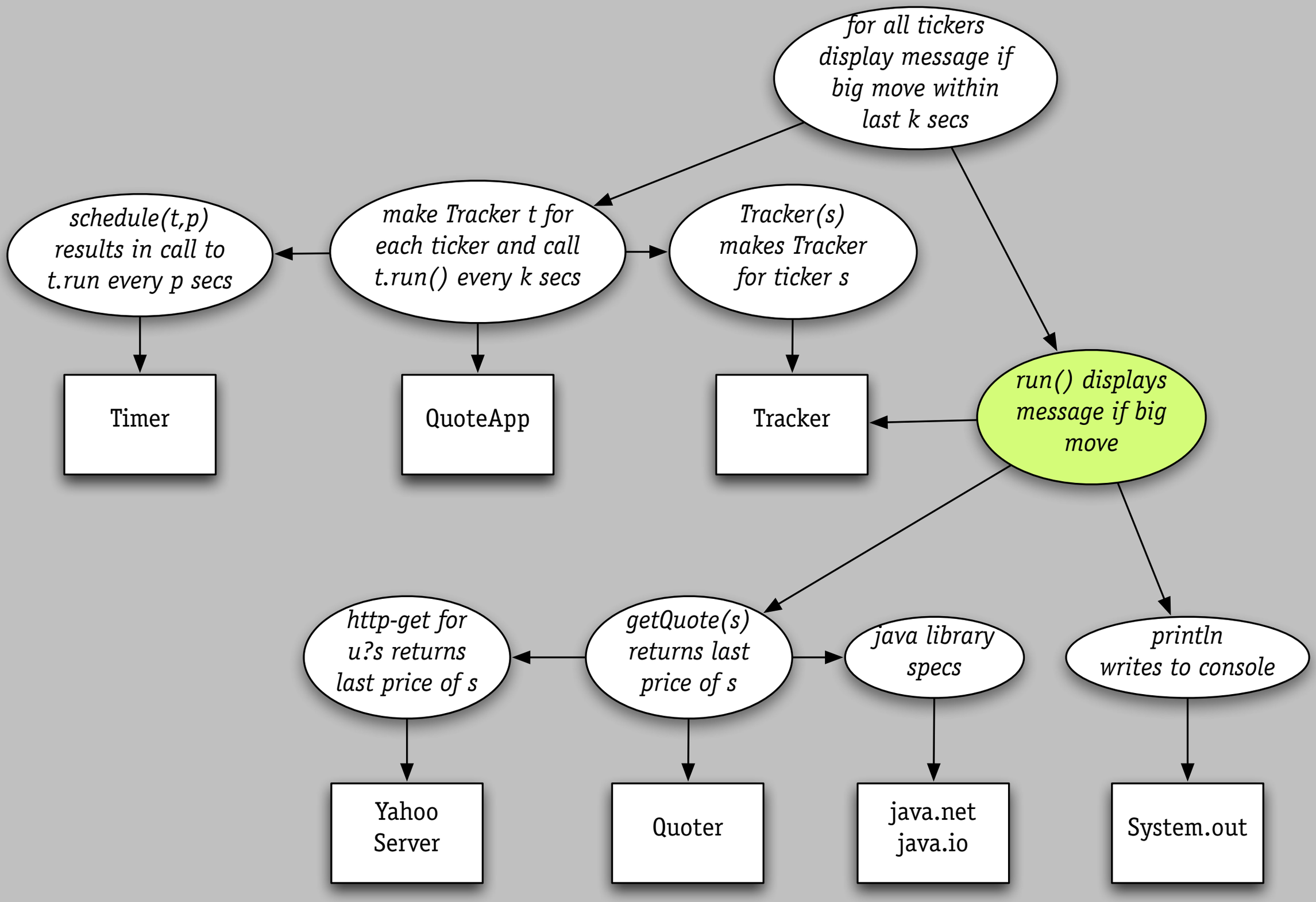
uses relation



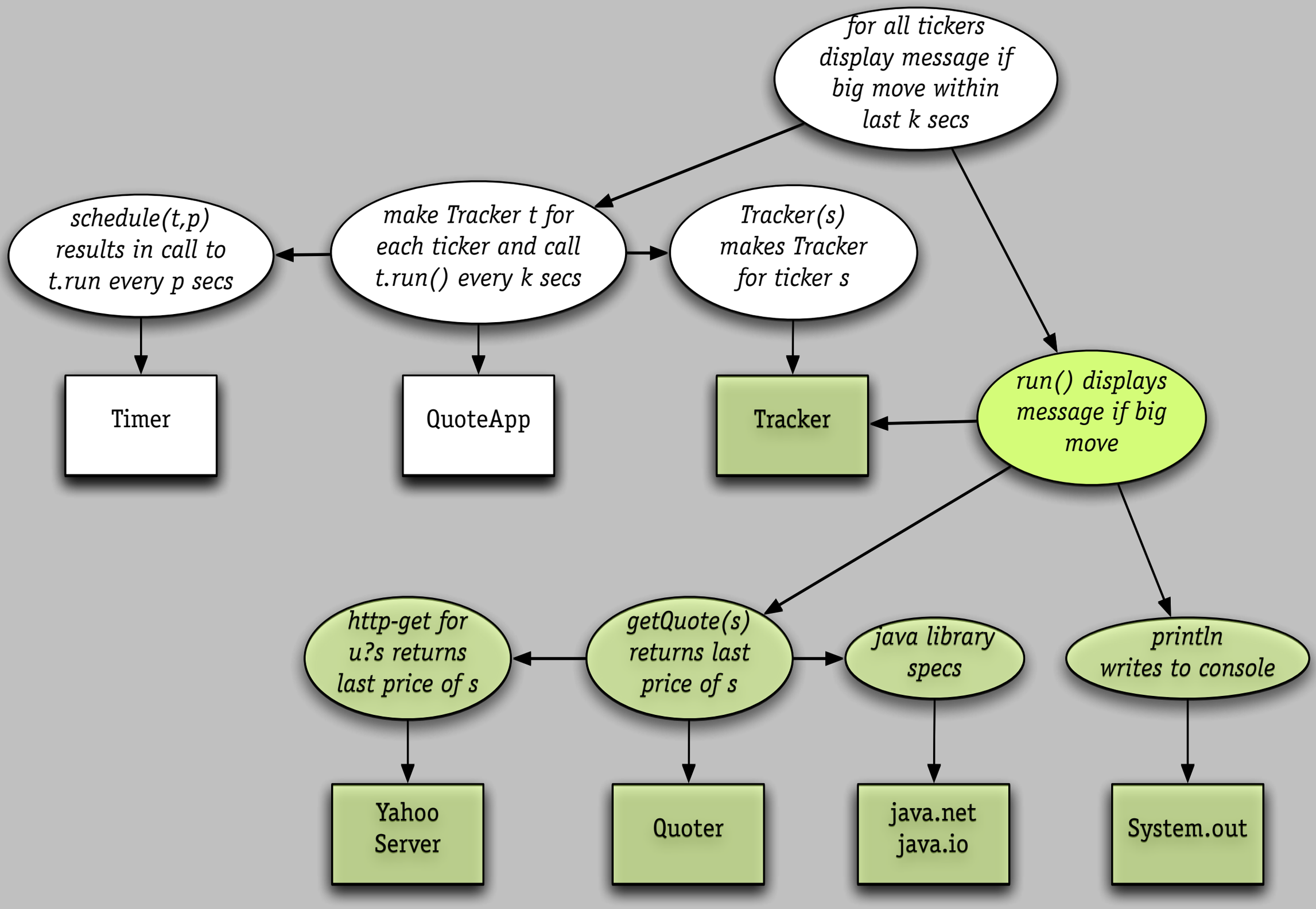
dependency diagram



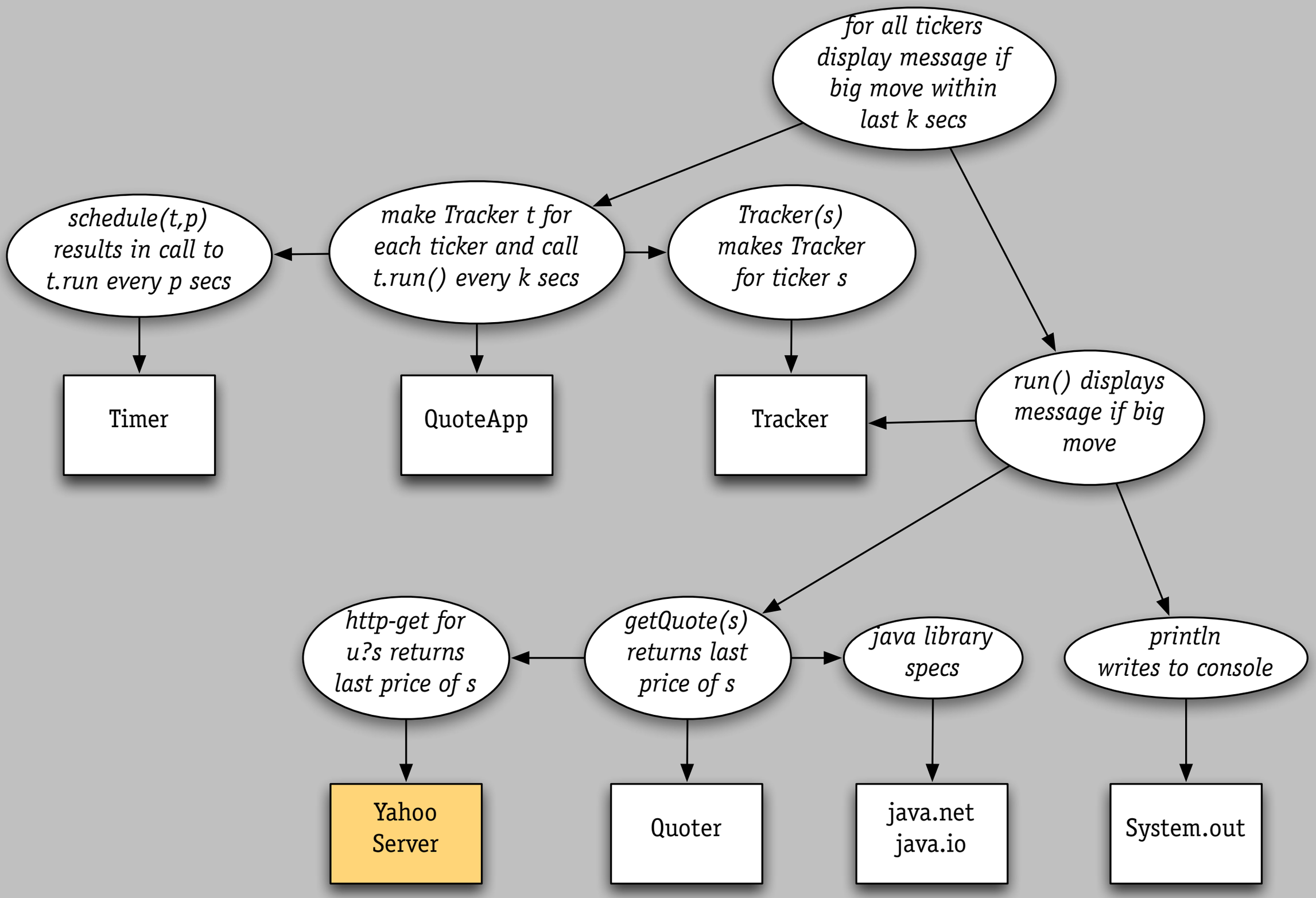
finding a property's support



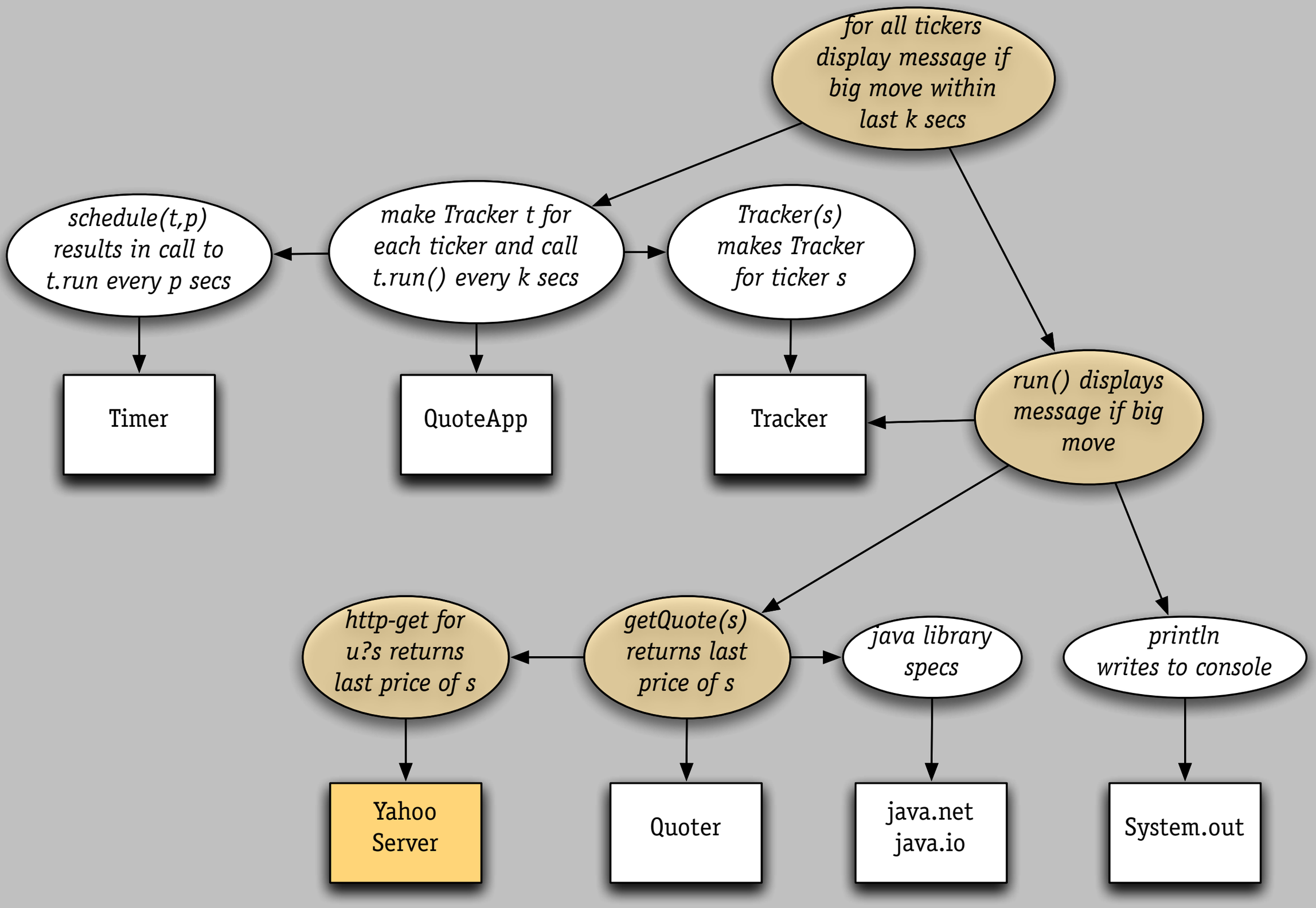
finding a property's support



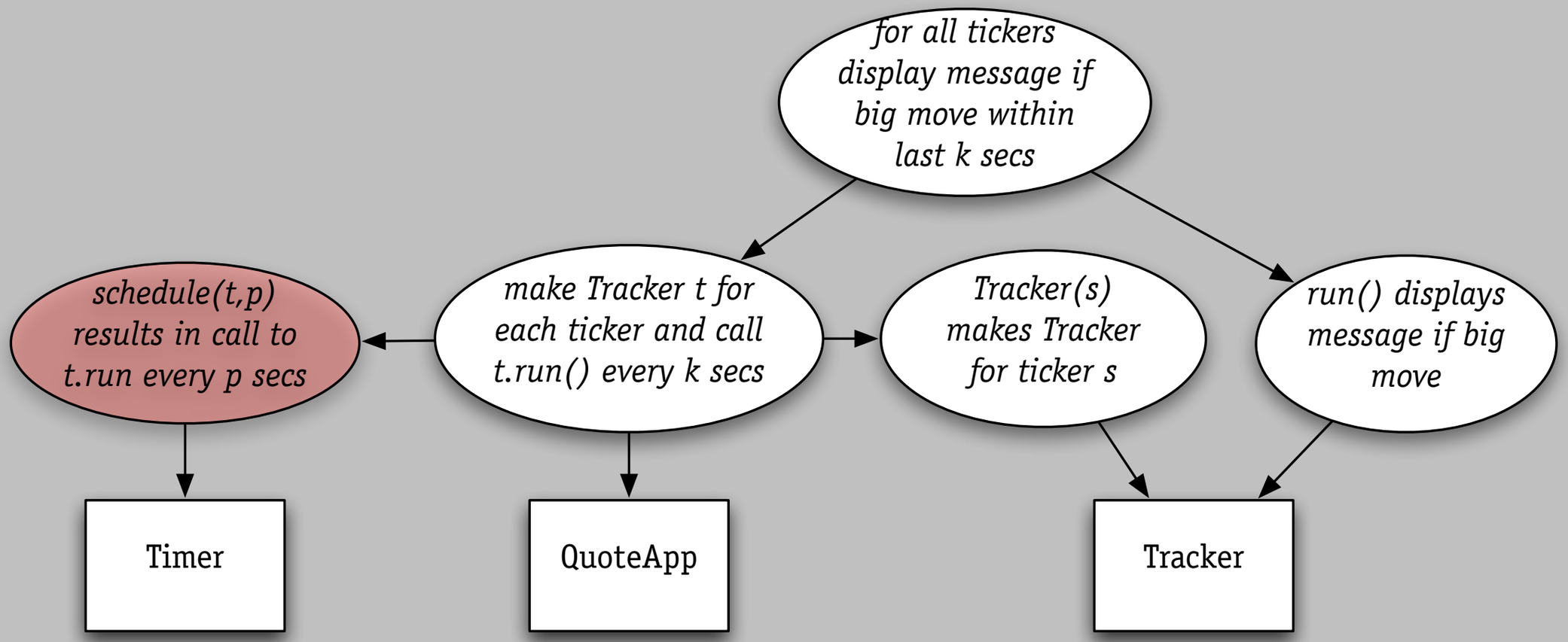
finding a component's impact



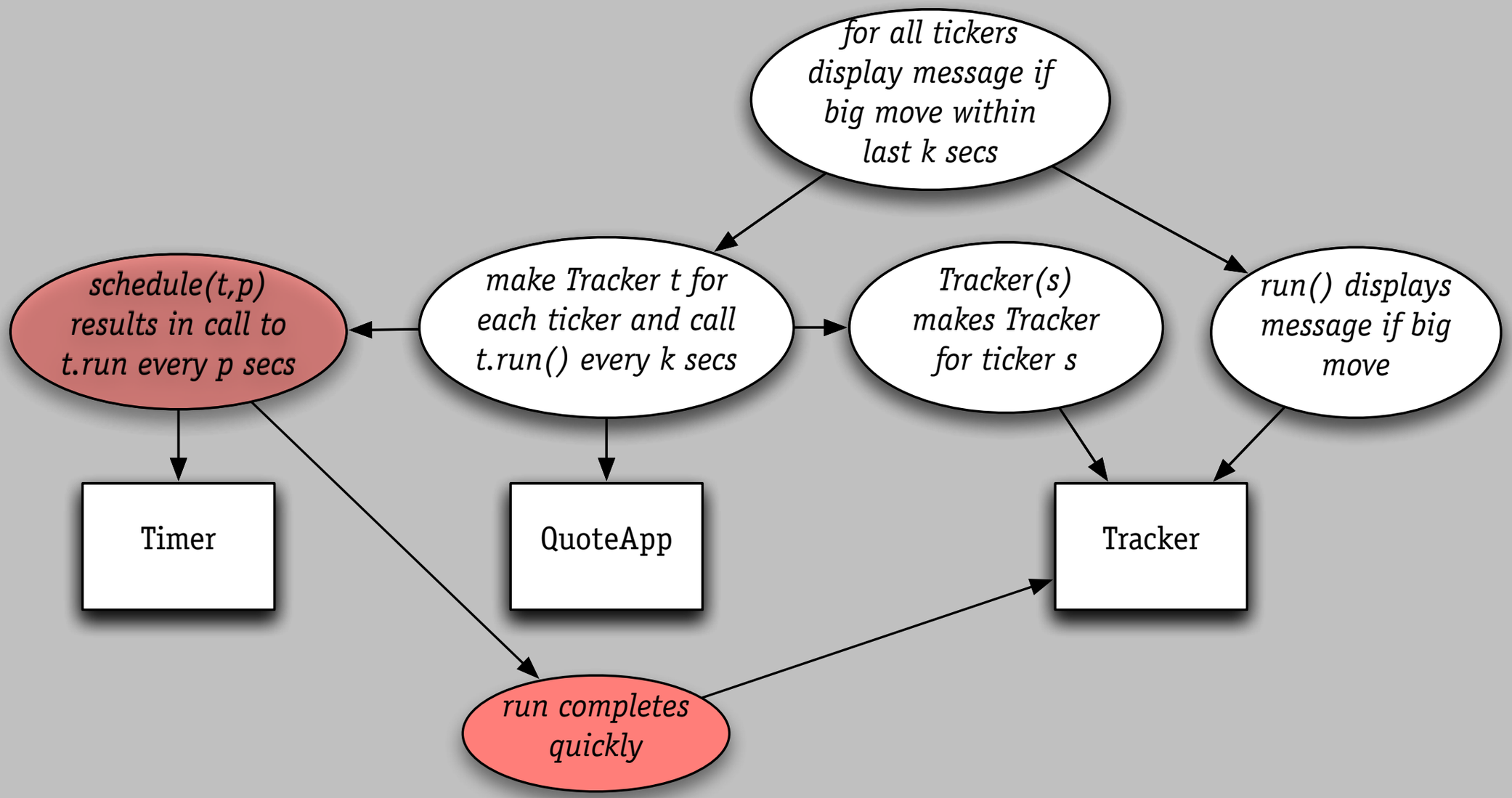
finding a component's impact



explaining a flaw



explaining a flaw



six failures, explained

apple file vault

securing files

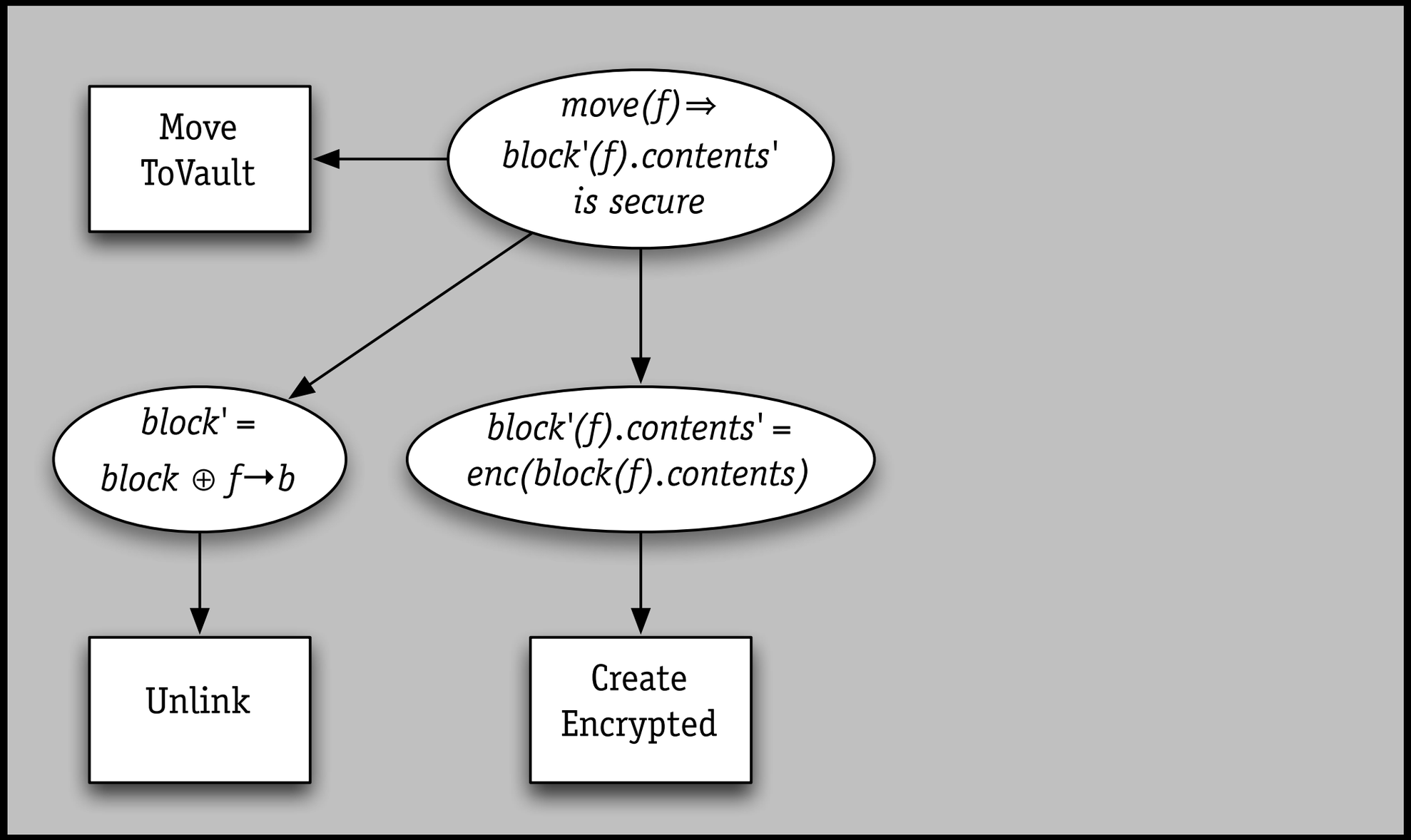
› make secure volume

› transfer files to it

what happens to old copies?

› unlinked but not erased!

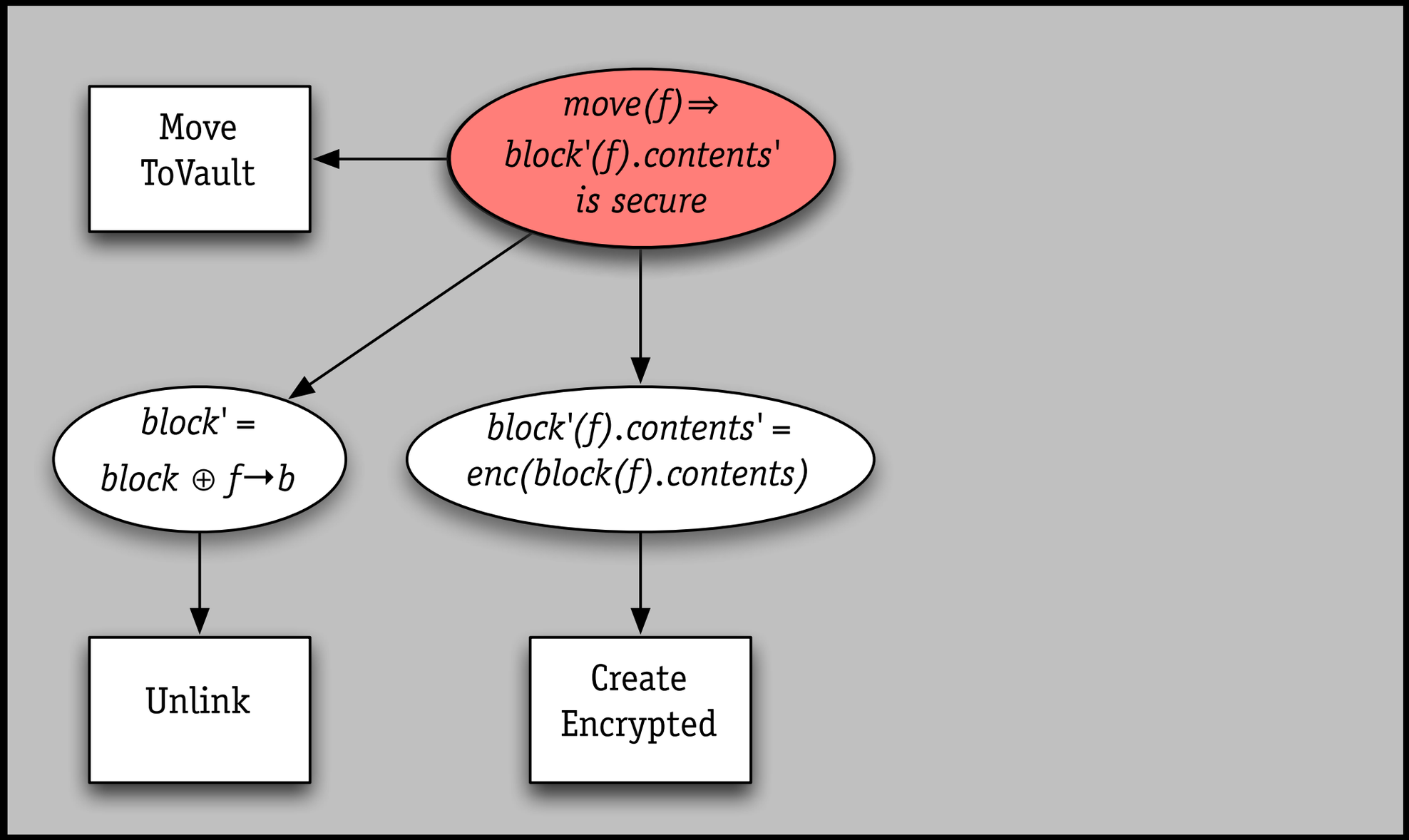
apple file vault



wrong property

from Simson Garfinkel, 2004

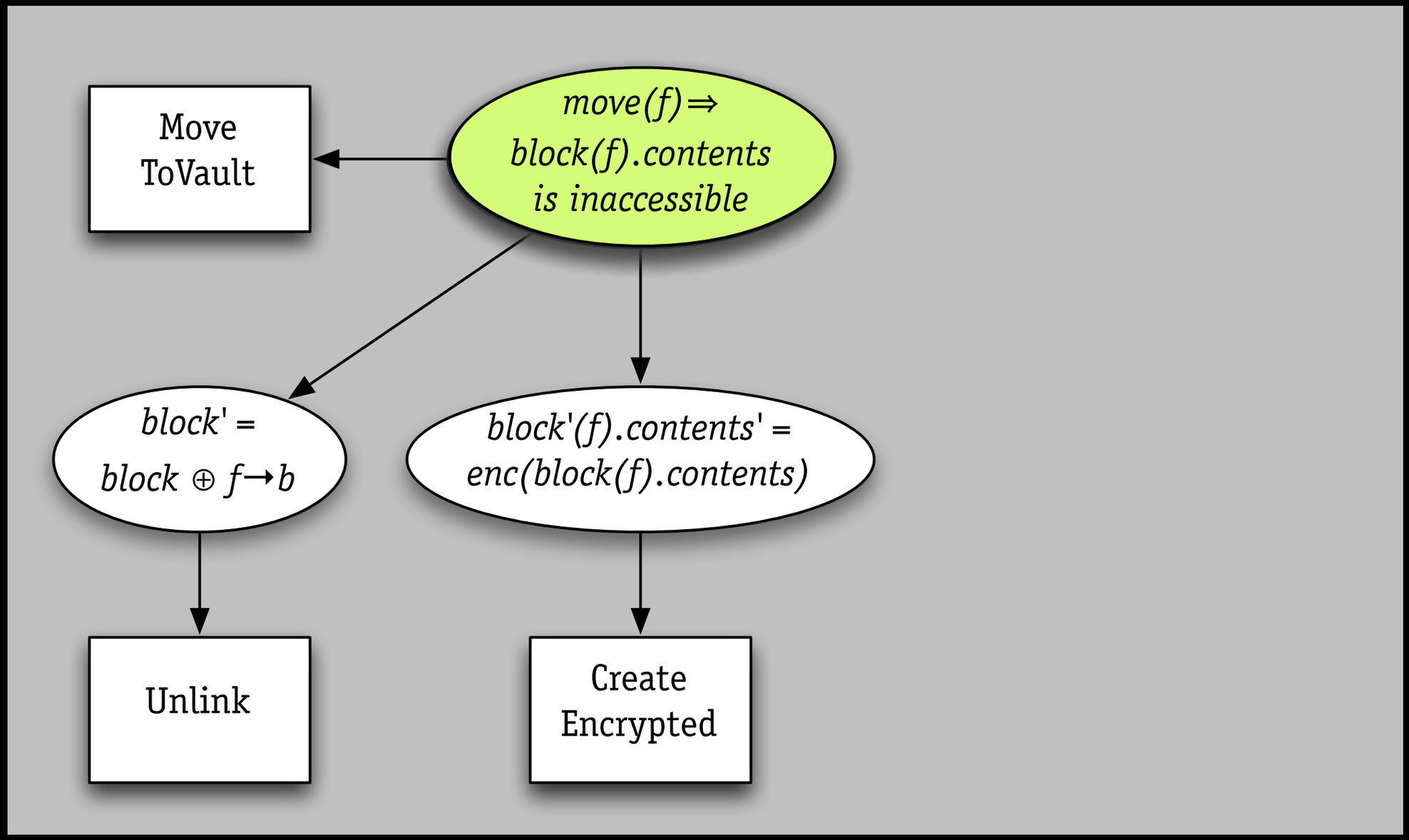
apple file vault



wrong property

from Simson Garfinkel, 2004

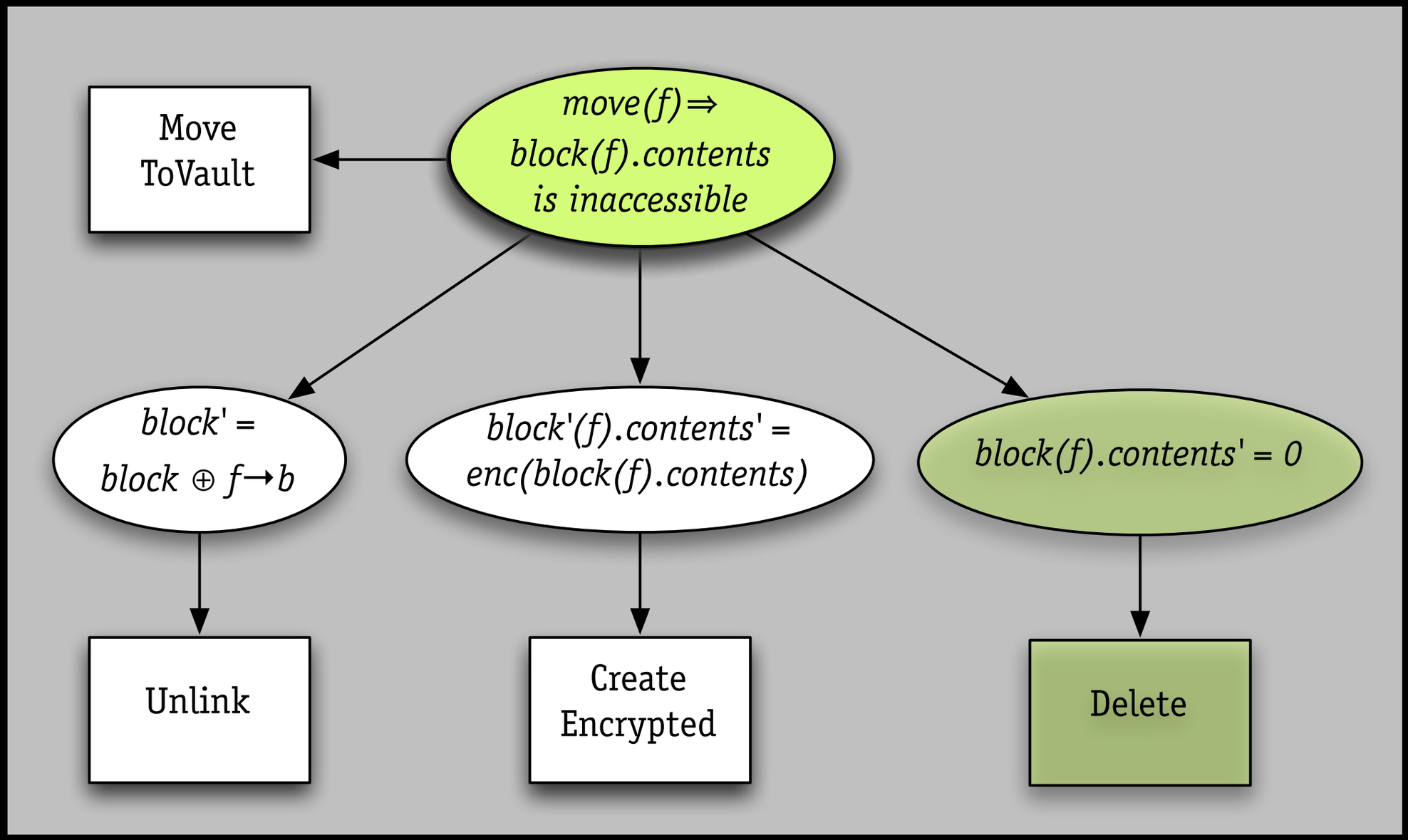
apple file vault



wrong property

from Simson Garfinkel, 2004

apple file vault



wrong property

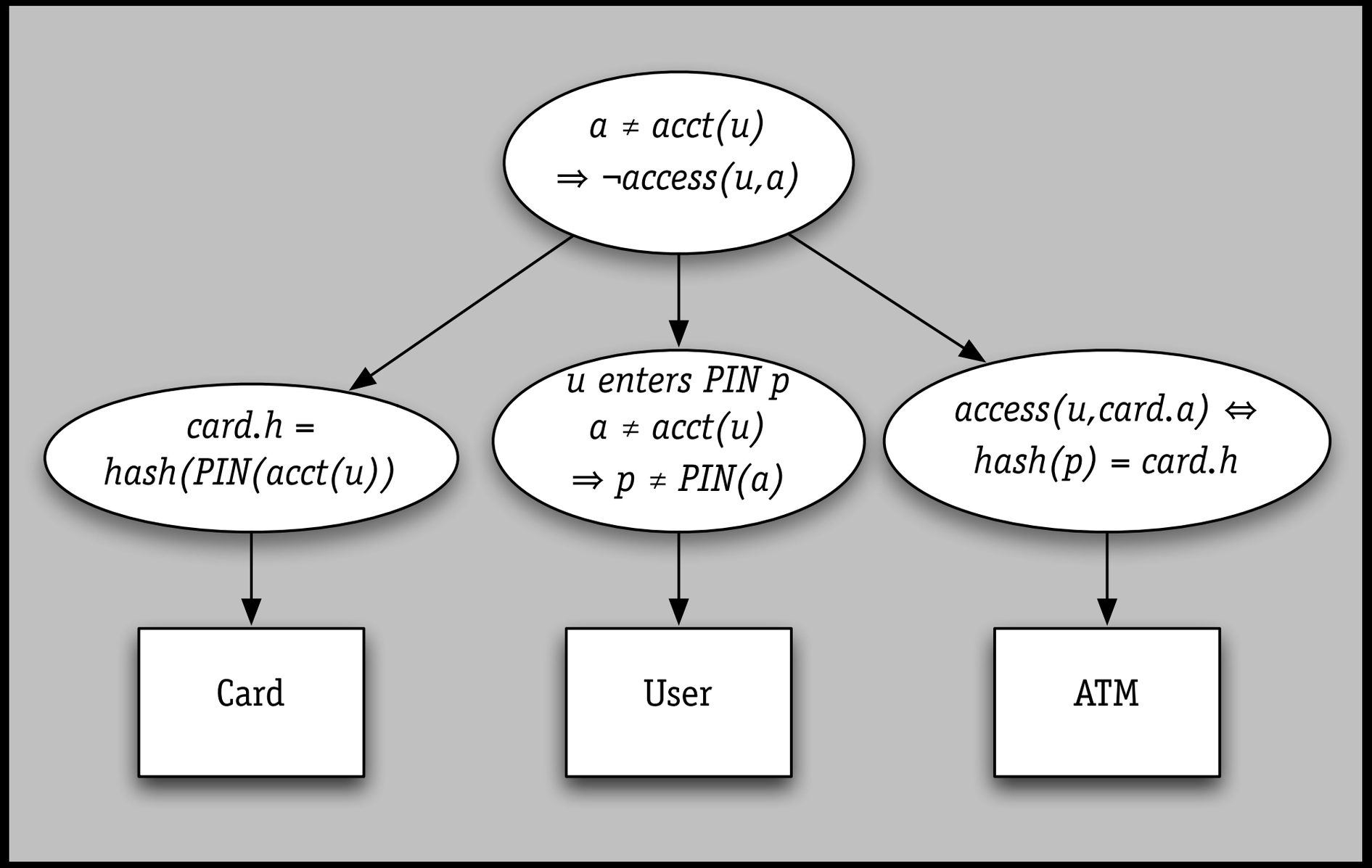
from Simson Garfinkel, 2004

insecure ATMs

a broken PIN scheme
› hash of PIN stored on card
› ATM just checks entered PIN against it

to access another account
› just change account number on card!

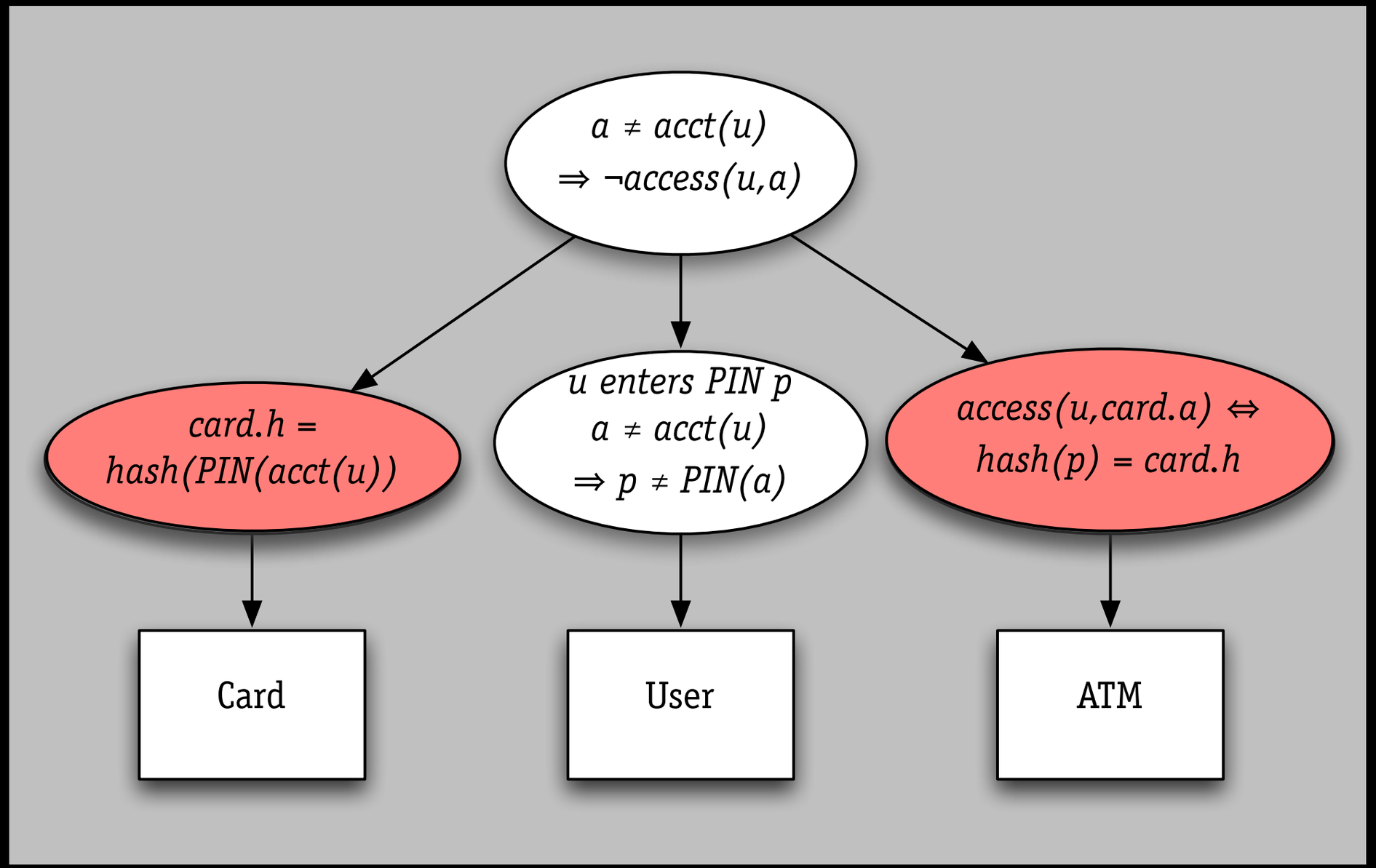
insecure ATMs



problem: bad analysis

from Ross Anderson, *Why Cryptosystems Fail, 1994*

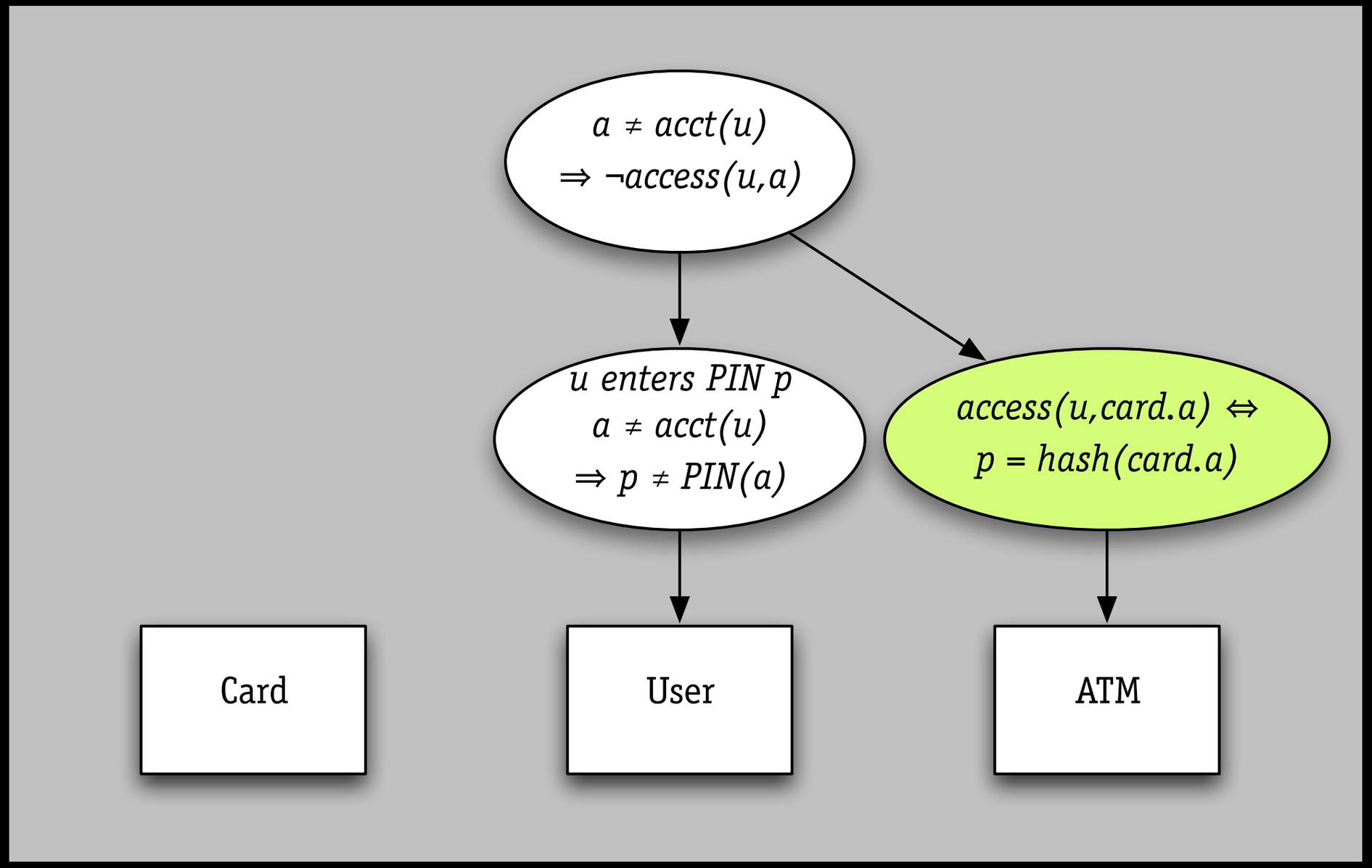
insecure ATMs



problem: bad analysis

from Ross Anderson, *Why Cryptosystems Fail, 1994*

insecure ATMs



problem: bad analysis

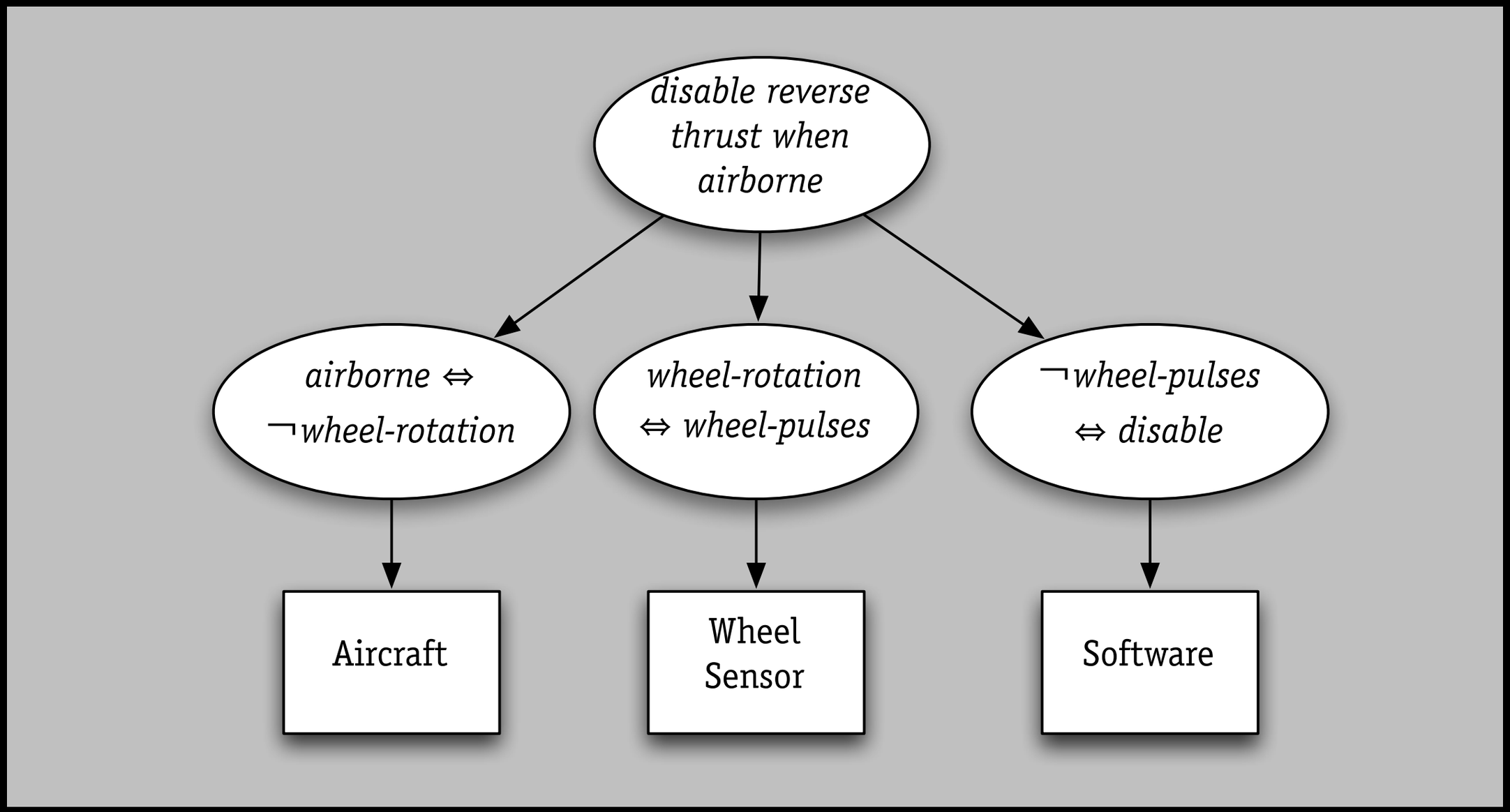
from Ross Anderson, *Why Cryptosystems Fail, 1994*

Airbus A320 (1993)

landing in Warsaw
› overrun runway
› pilot & passenger died

explanation
› aquaplaned, so no wheel rotation
› reverse thrust was disabled for 9s

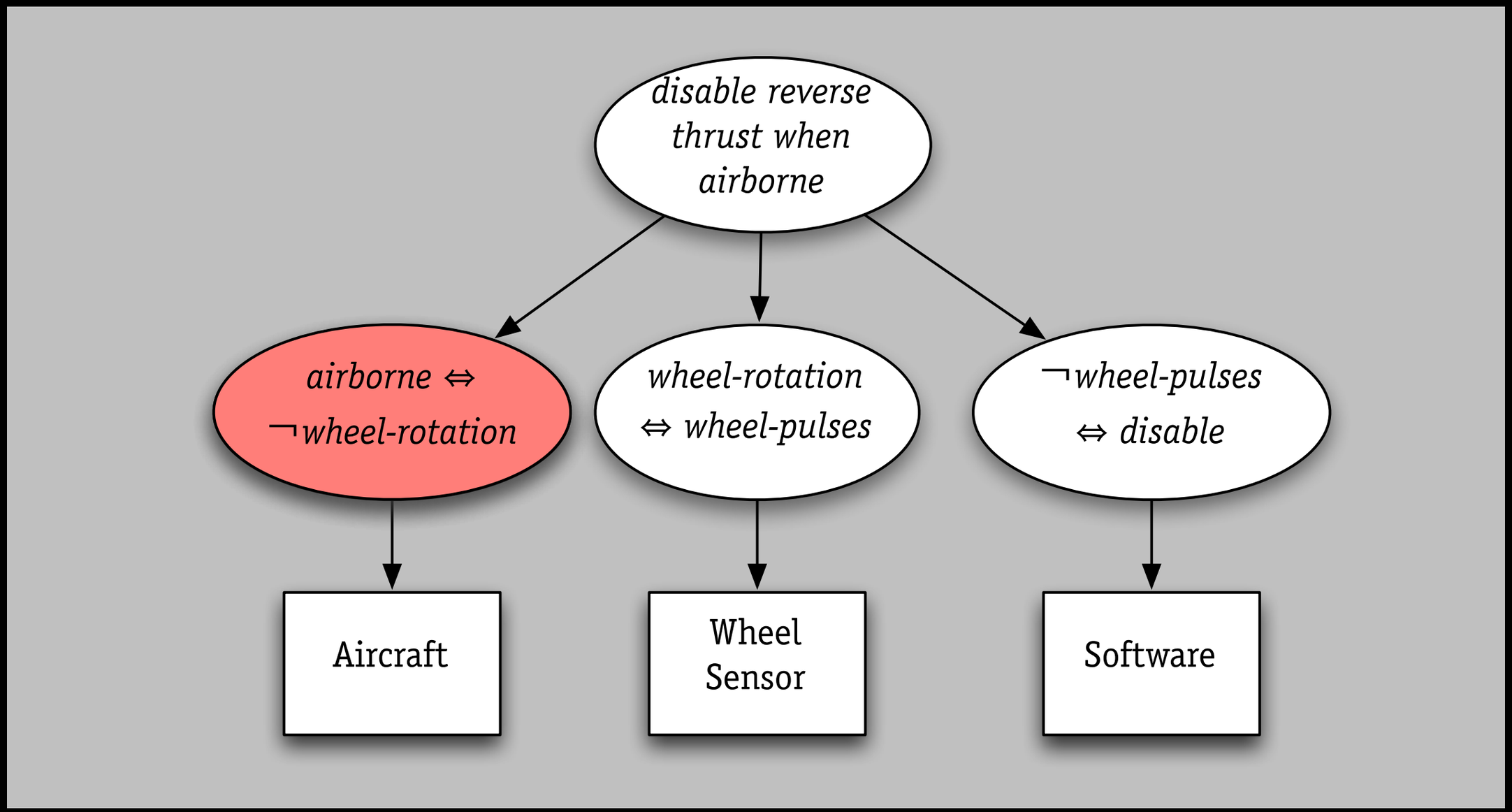
Airbus A320 (1993)



problem: incorrect environmental assumption

from Michael Jackson, Peter Ladkin

Airbus A320 (1993)



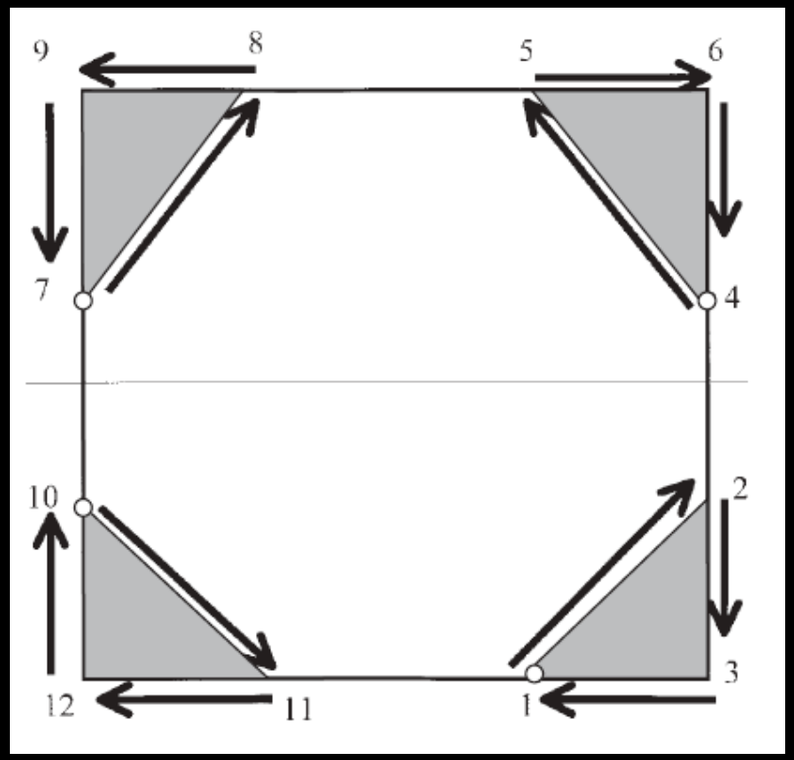
problem: incorrect environmental assumption

from Michael Jackson, Peter Ladkin

Panama City (2001)

radiation treatment planning software
overexposes 20, killing at least 9

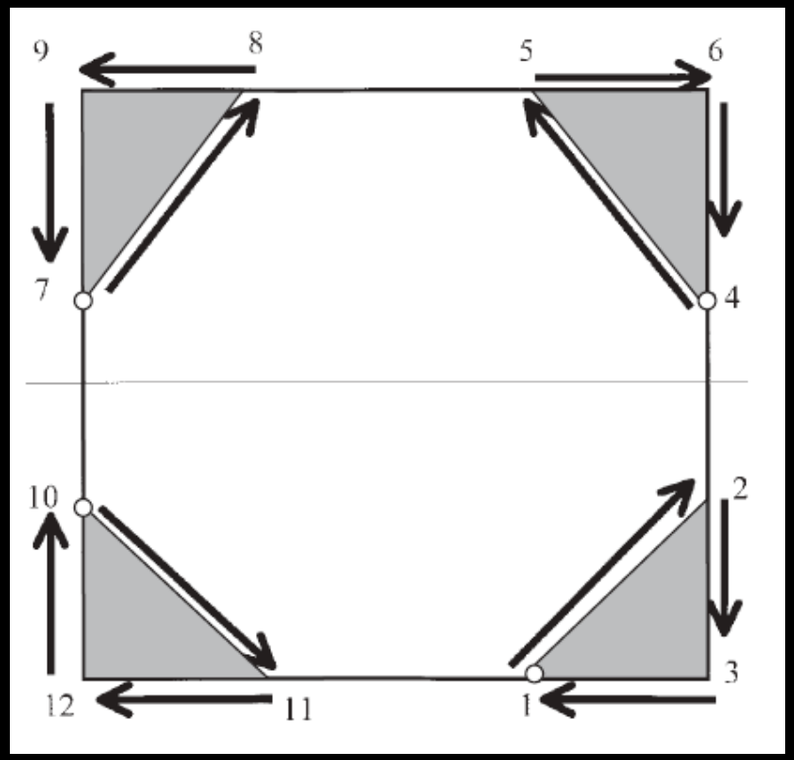
Panama City (2001)



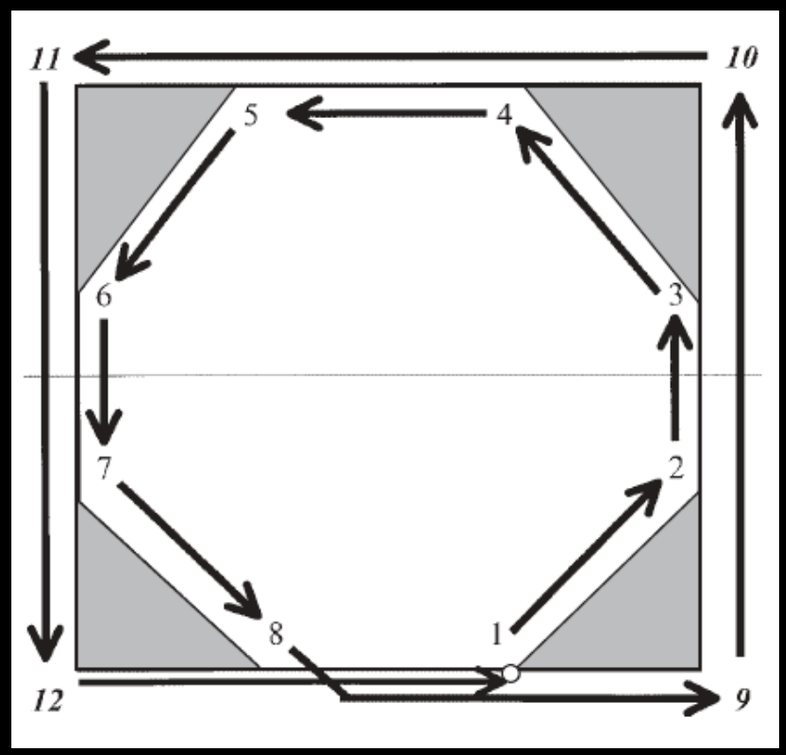
dose = D

radiation treatment planning software
overexposes 20, killing at least 9

Panama City (2001)



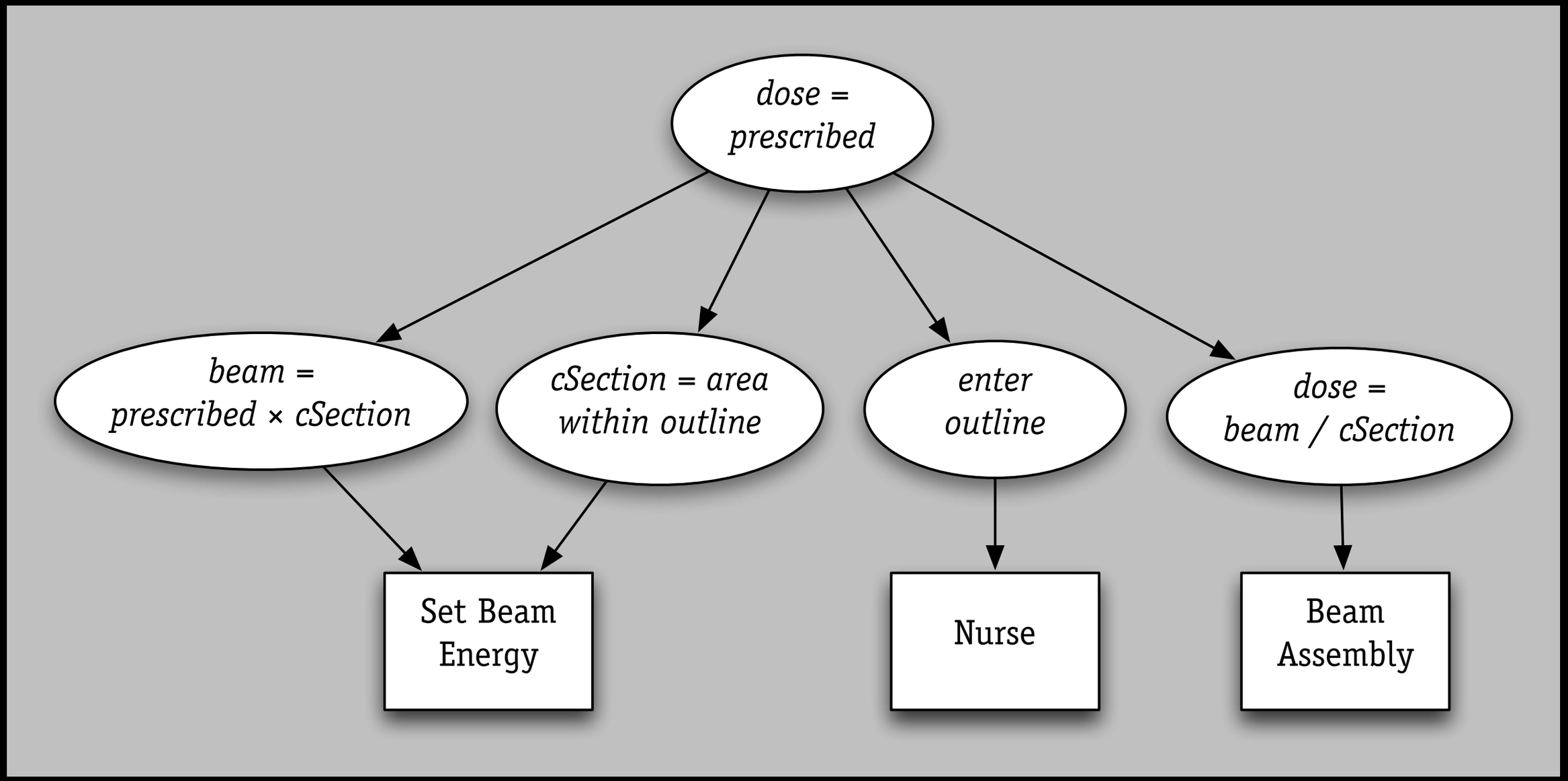
dose = D



dose = 2D

radiation treatment planning software
overexposes 20, killing at least 9

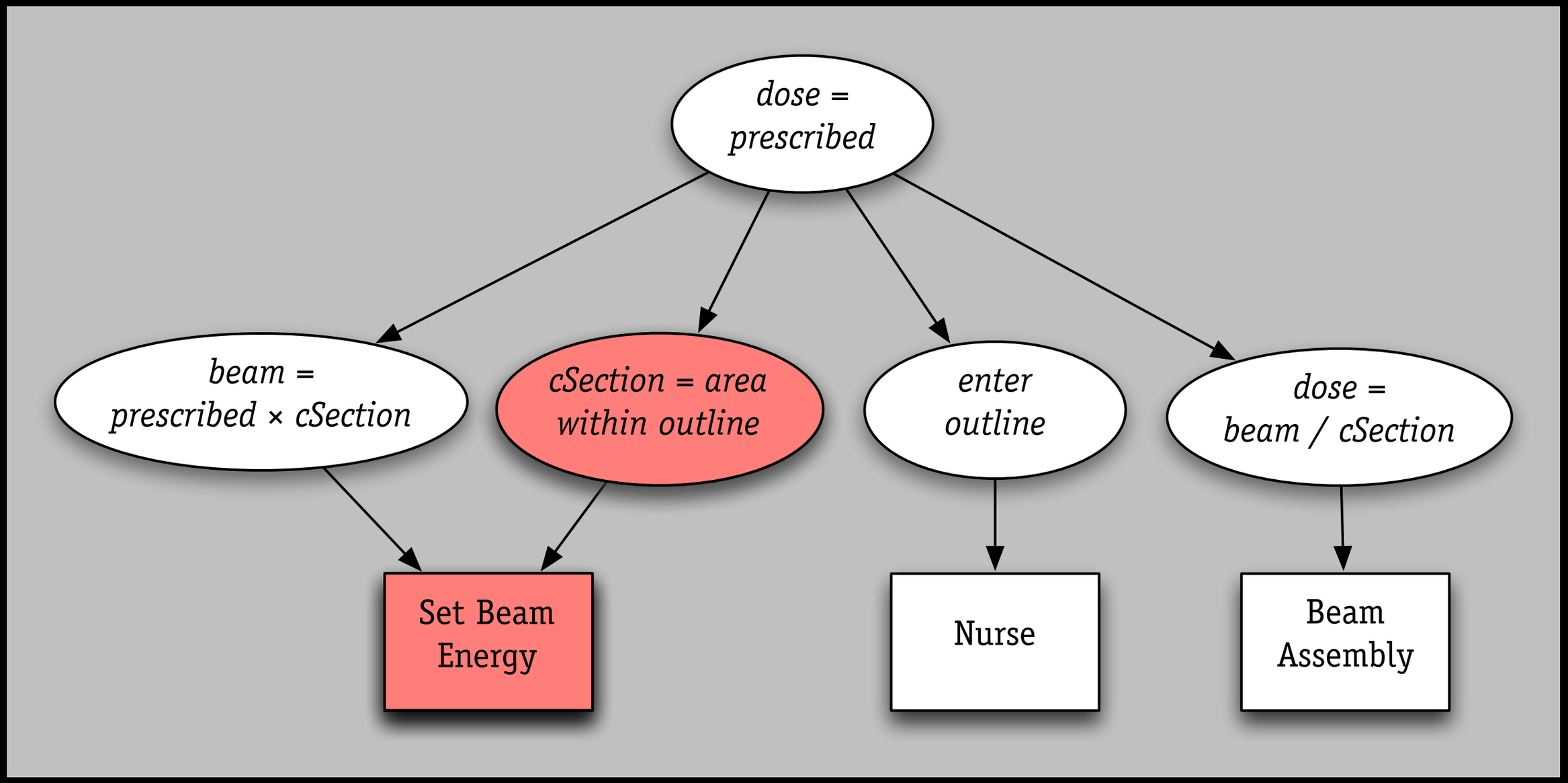
Panama Radiotherapy, 2001



problem: component fails to meet spec

from IAEA Investigation, 2001

Panama Radiotherapy, 2001



problem: component fails to meet spec

from IAEA Investigation, 2001

Given [the input] that was given, our system calculated the correct amount, the correct dose. It was an unexpected result. And, if [the staff in Panama] had checked, they would have found an unexpected result.

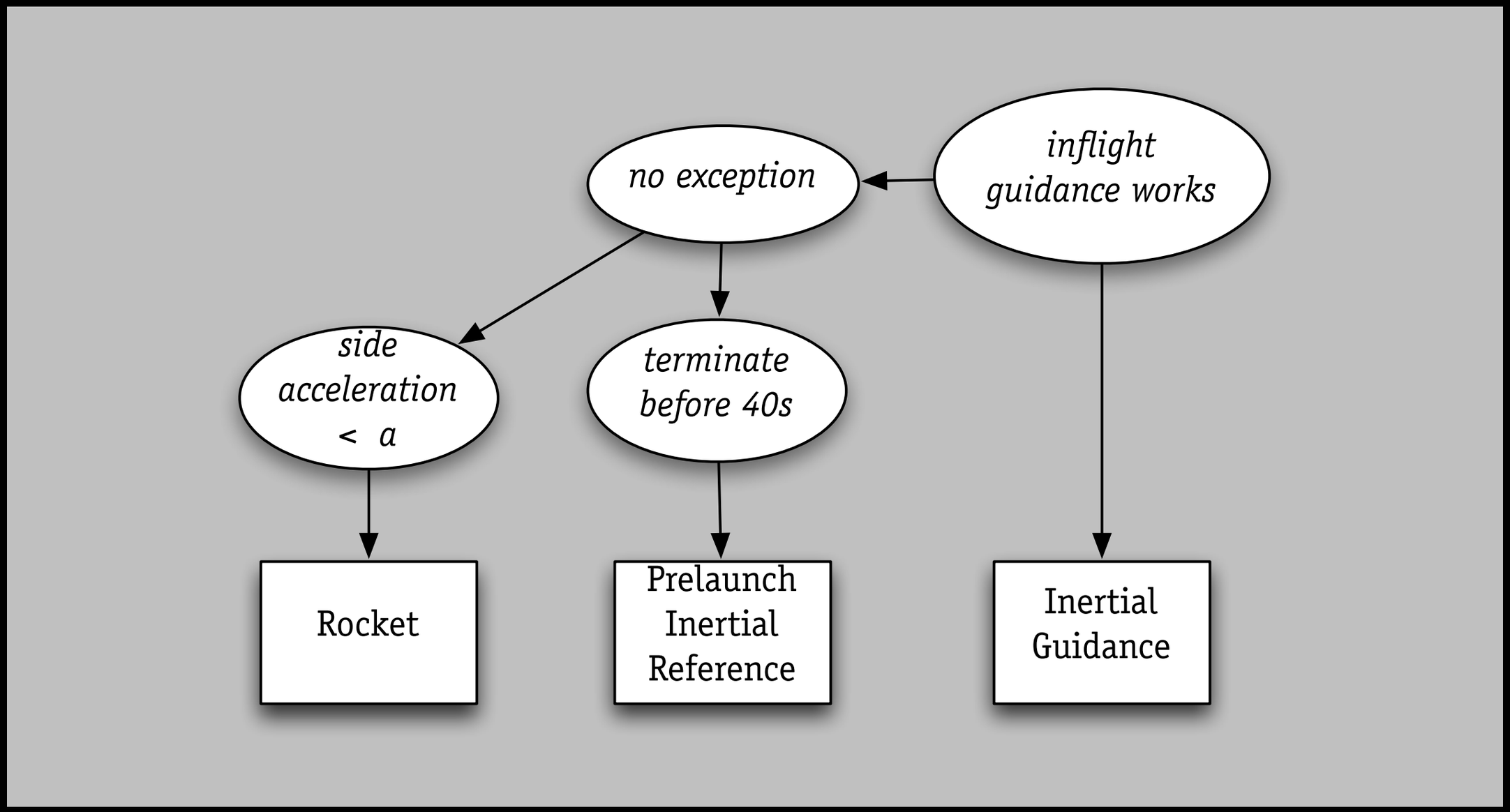
Mick Conley, Multidata

Ariane 5 (1996)

first test flight
› self destructs
› $400m loss

explanation
› prelaunch routine throws numeric overflow exception

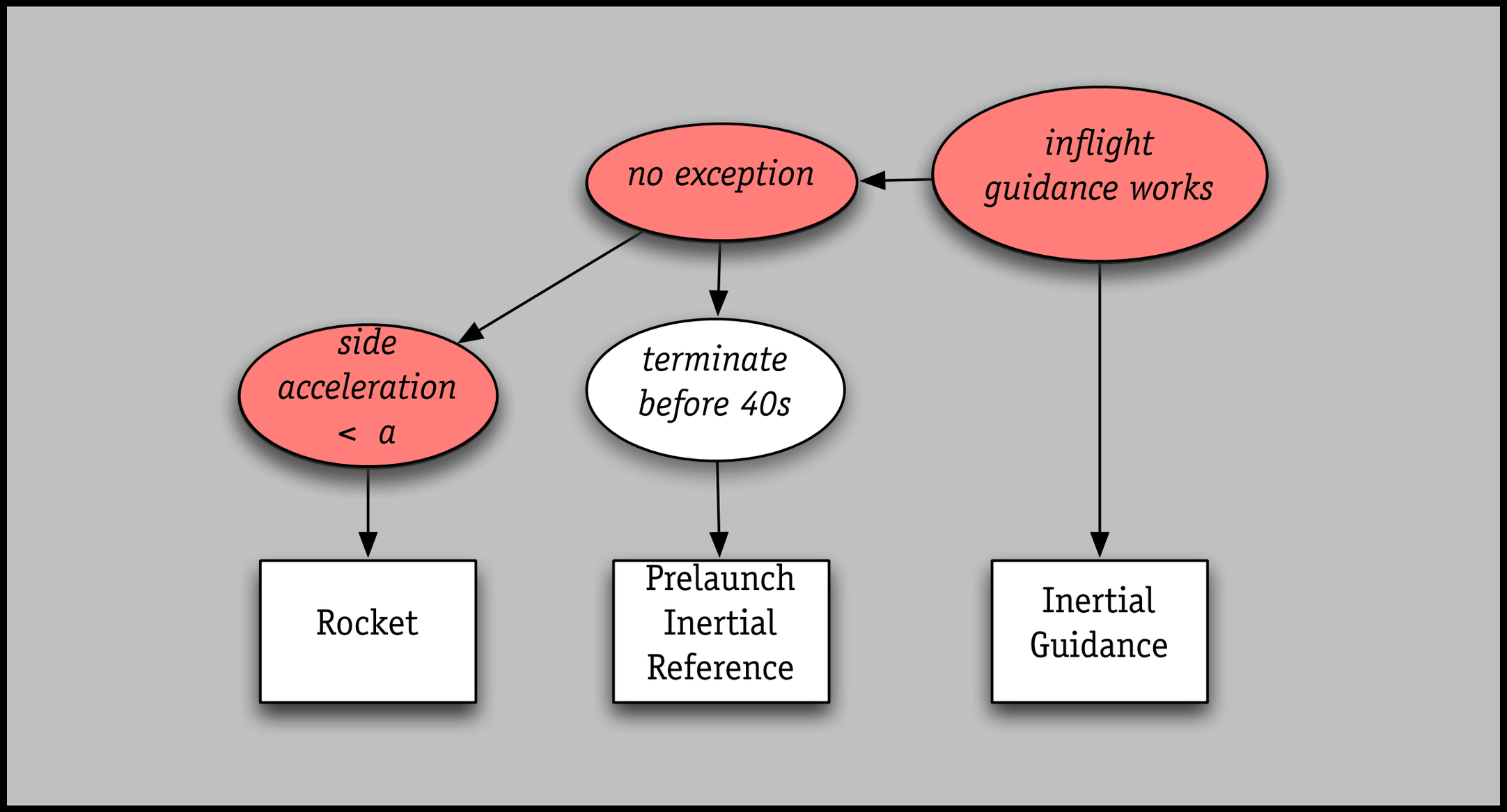
Ariane 5, 1996



problem: neglected coupling, incorrect assumptions

from report of inquiry board, J.L. Lions

Ariane 5, 1996



problem: neglected coupling, incorrect assumptions

from report of inquiry board, J.L. Lions

AT&T outage (1990)

failure in 5ESS switch

› for 9 hours

› 148 calls made

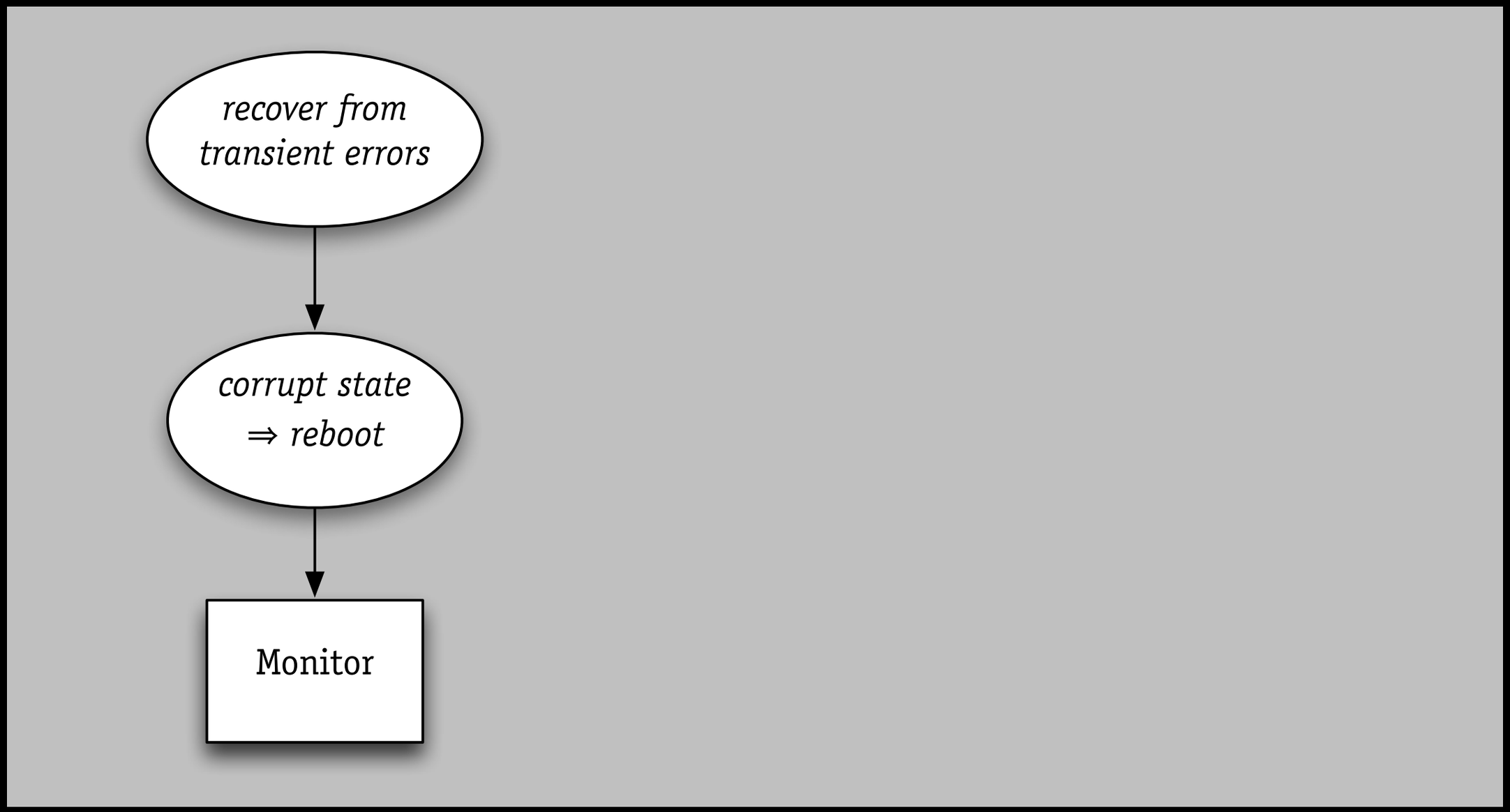
› about 50% dropped

explanation

› bug in recent upgrade

› caused knock-on crashes

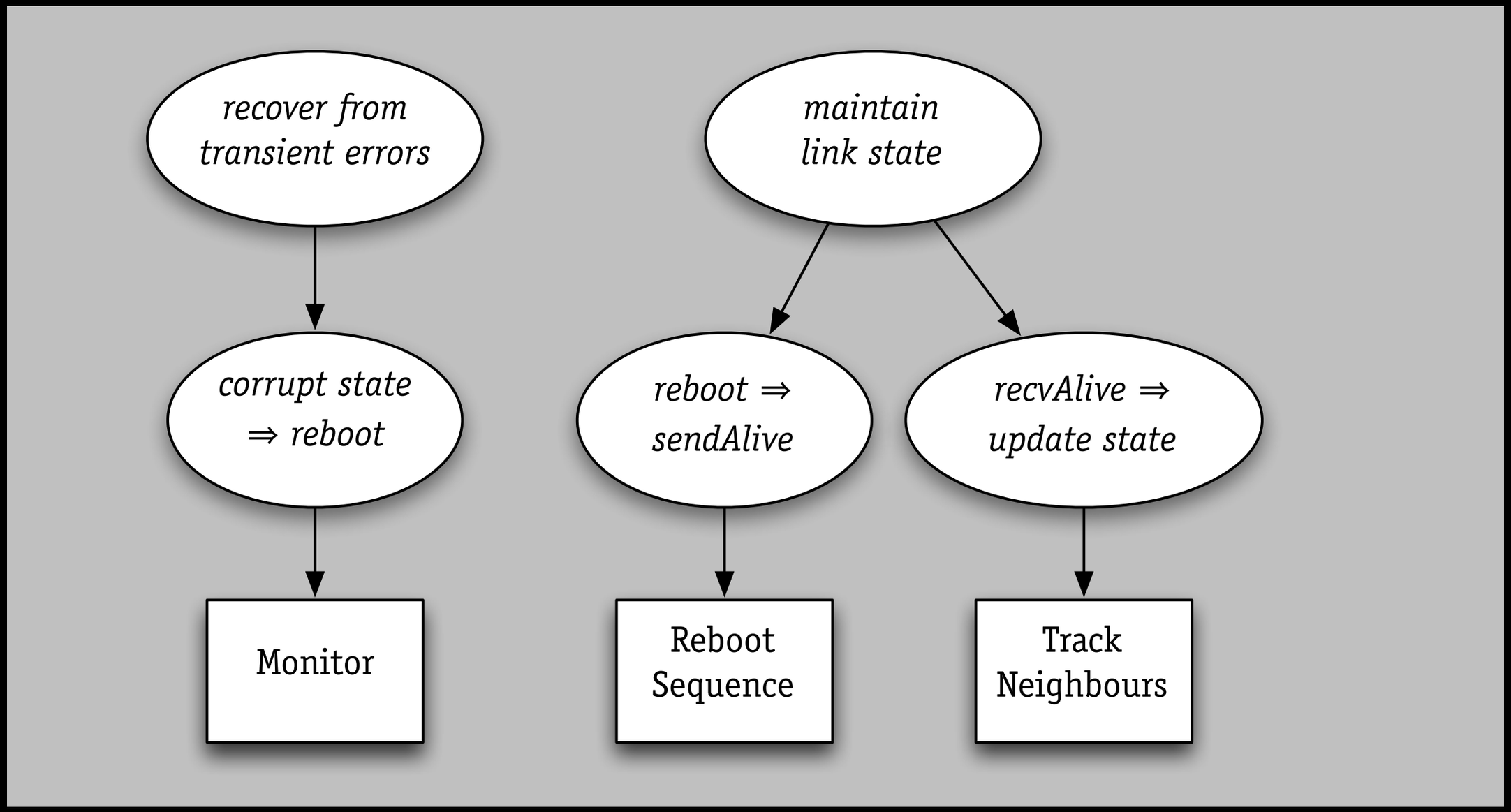
AT&T outage (1990)



problem: feature interaction

from RISKS Forum

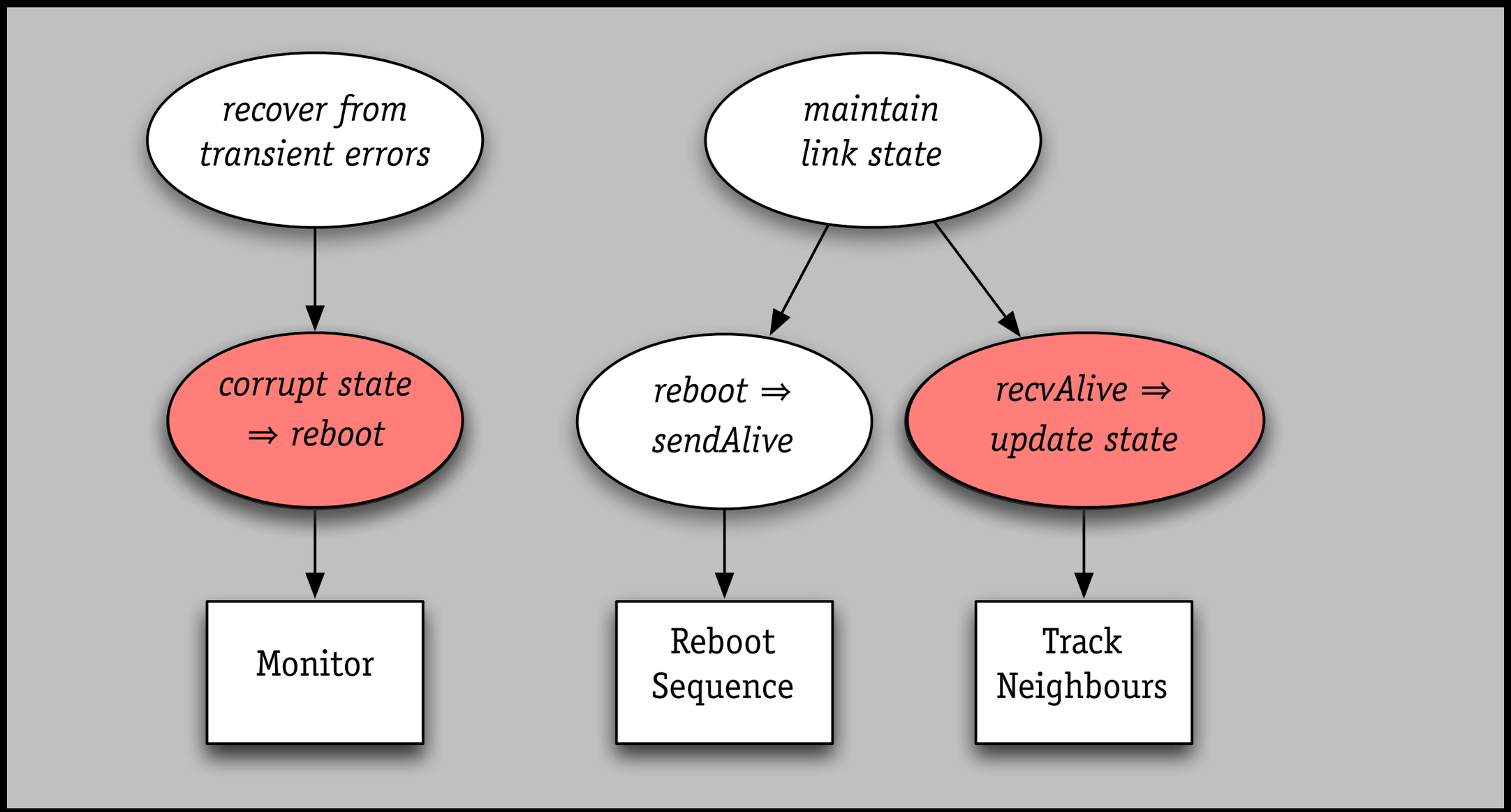
AT&T outage (1990)



problem: feature interaction

from RISKS Forum

AT&T outage (1990)



problem: feature interaction

from RISKS Forum

plus ça change...

Phone-company technicians traced the problem to a single 'failure of logic' in the computer programs that route calls through the AT&T network.

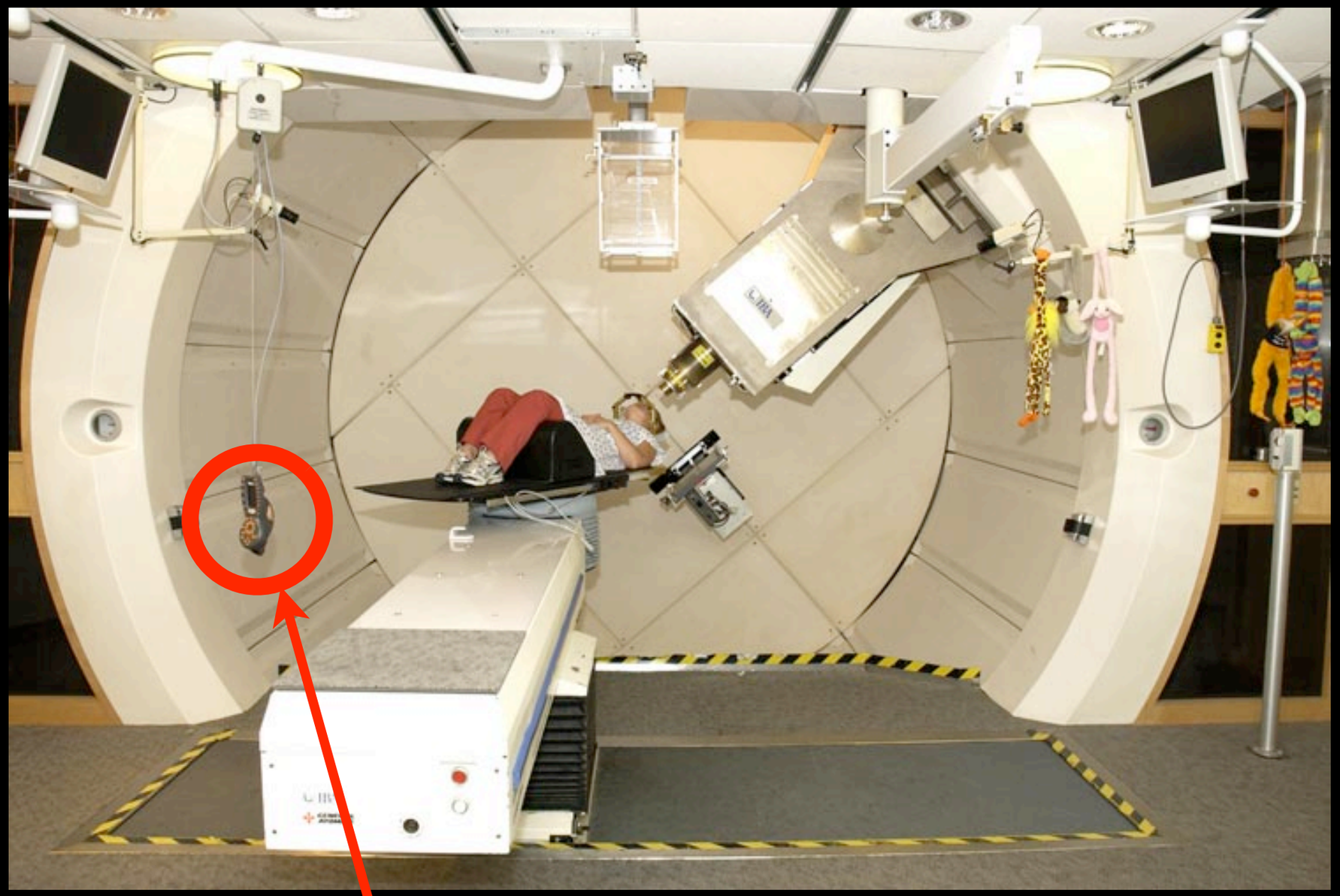
AT&T Network Outage, 1990

We've now determined that message corruption was the cause of the server-to-server communication problems ... a handful of messages ... had a single bit corrupted

Amazon S3 Outage, 2009

case study:
proton therapy

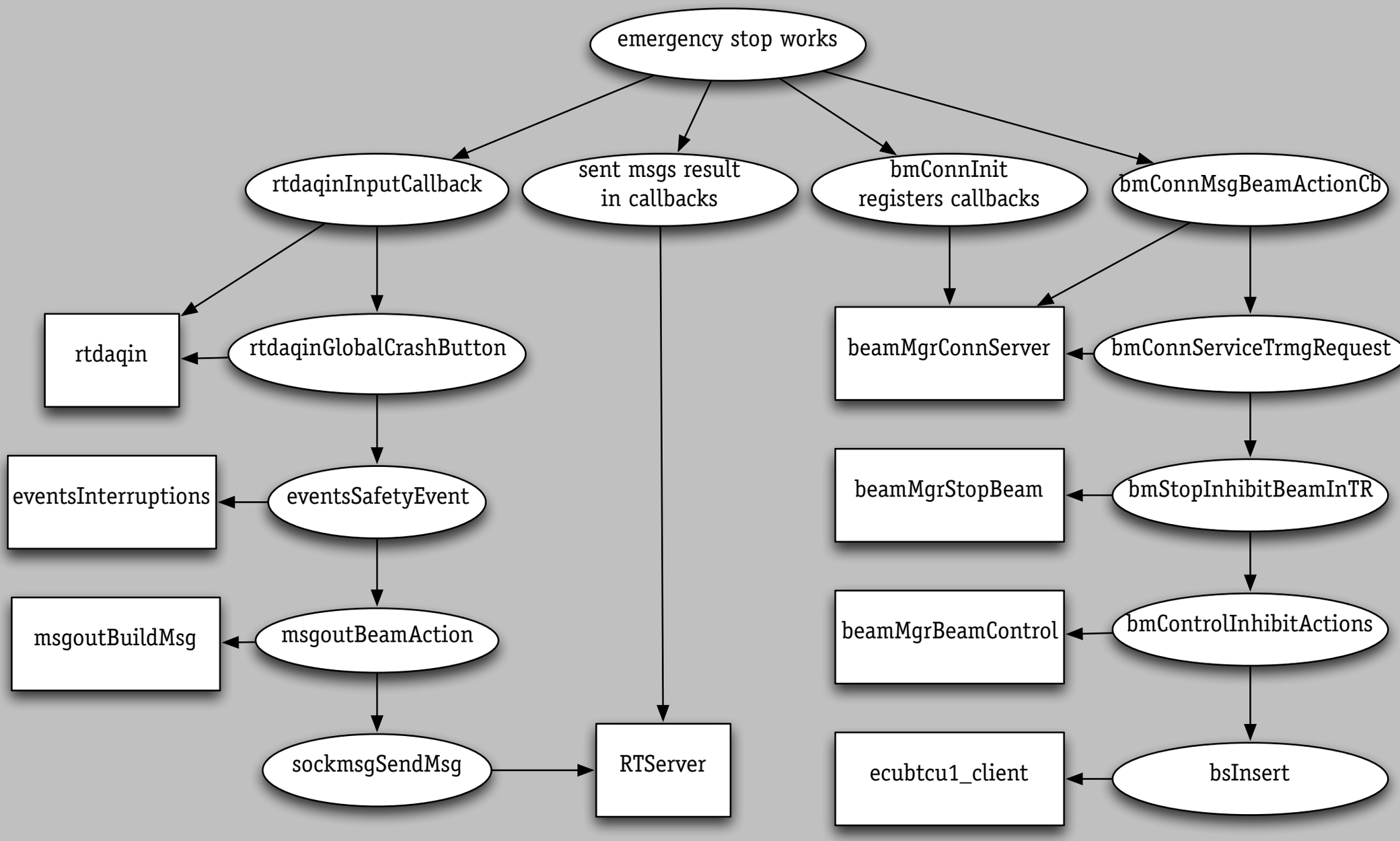
proton therapy



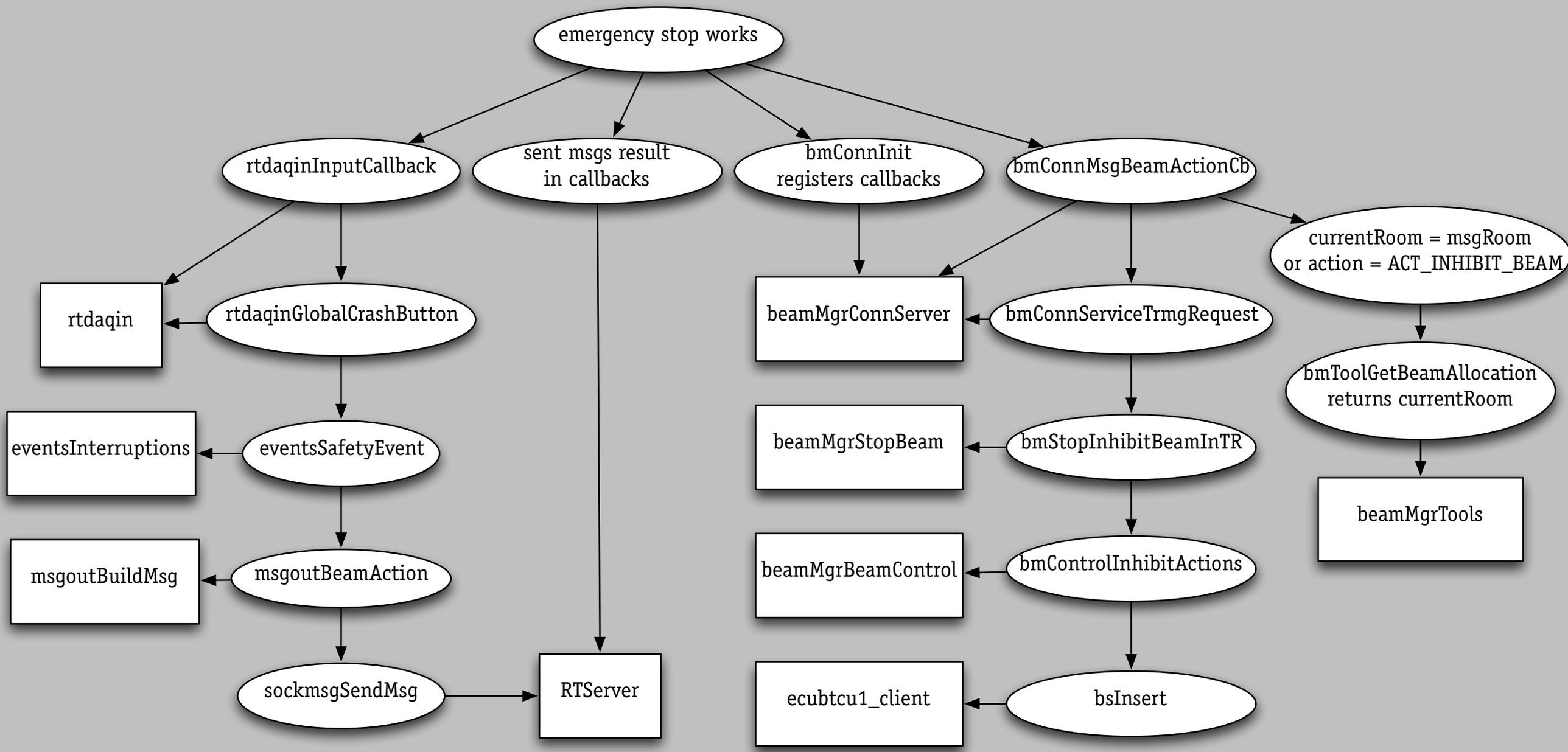
hand pendant with stop button

tracing emergency stop

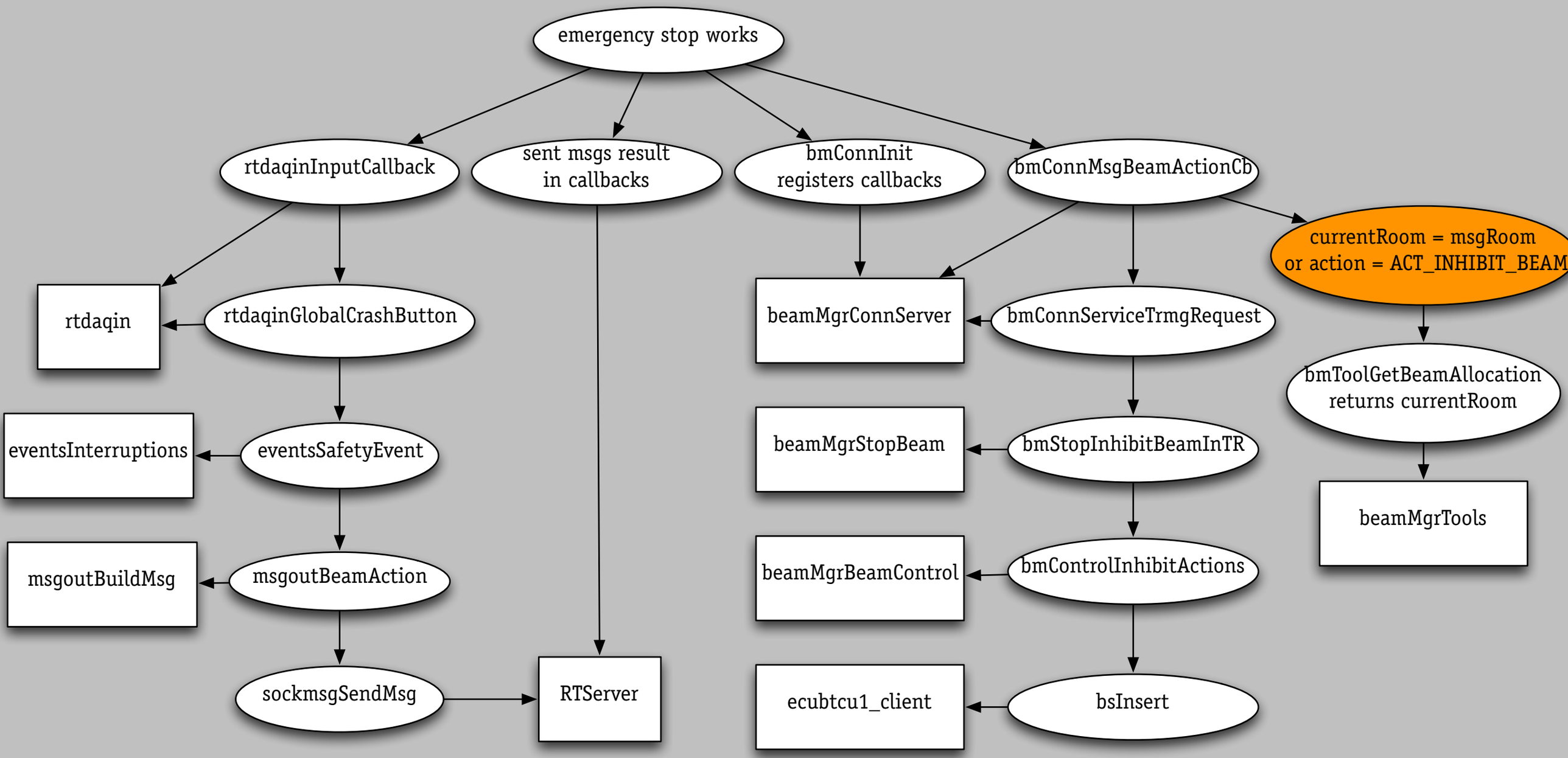
tracing emergency stop



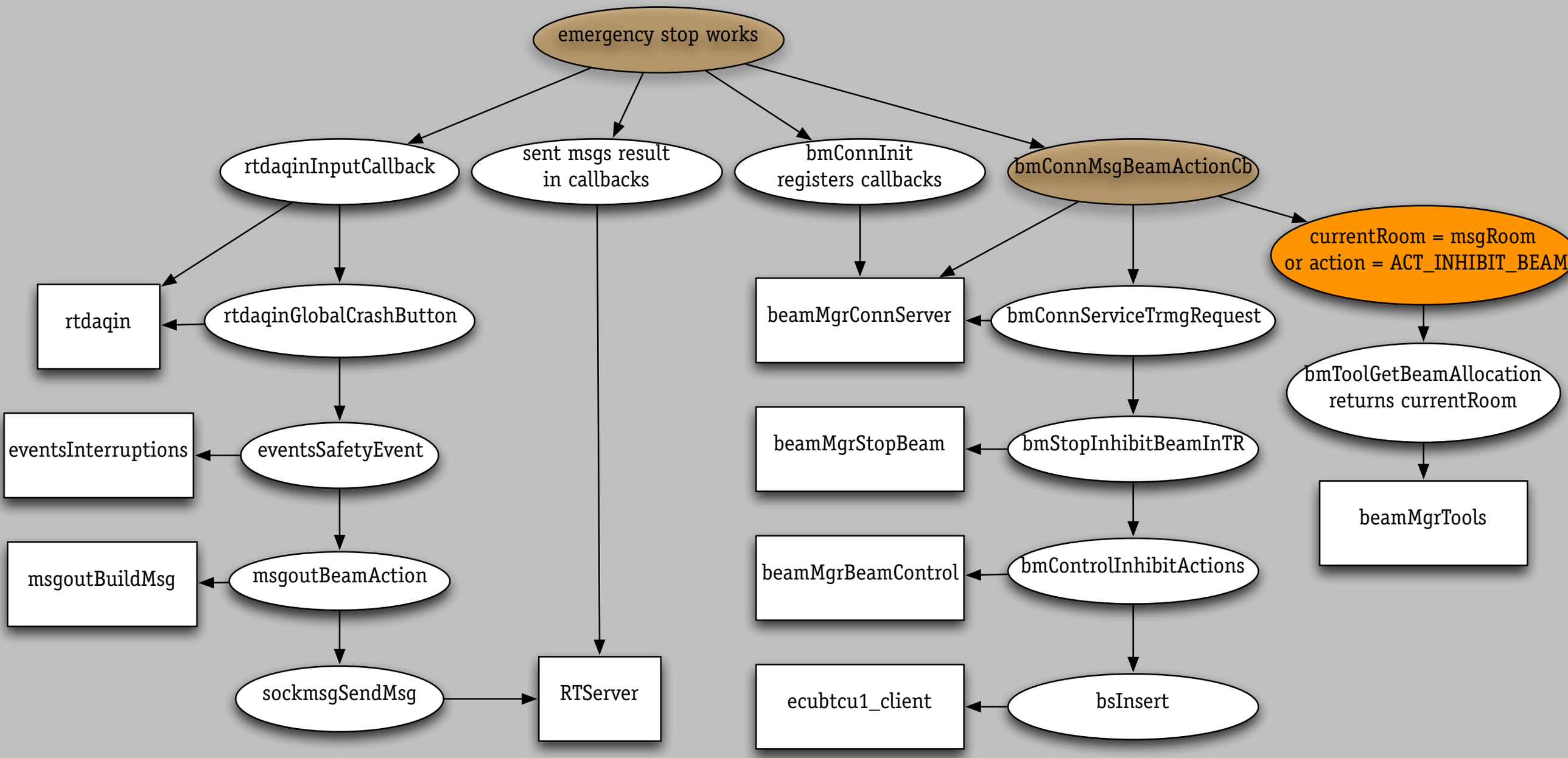
tracing emergency stop



tracing emergency stop

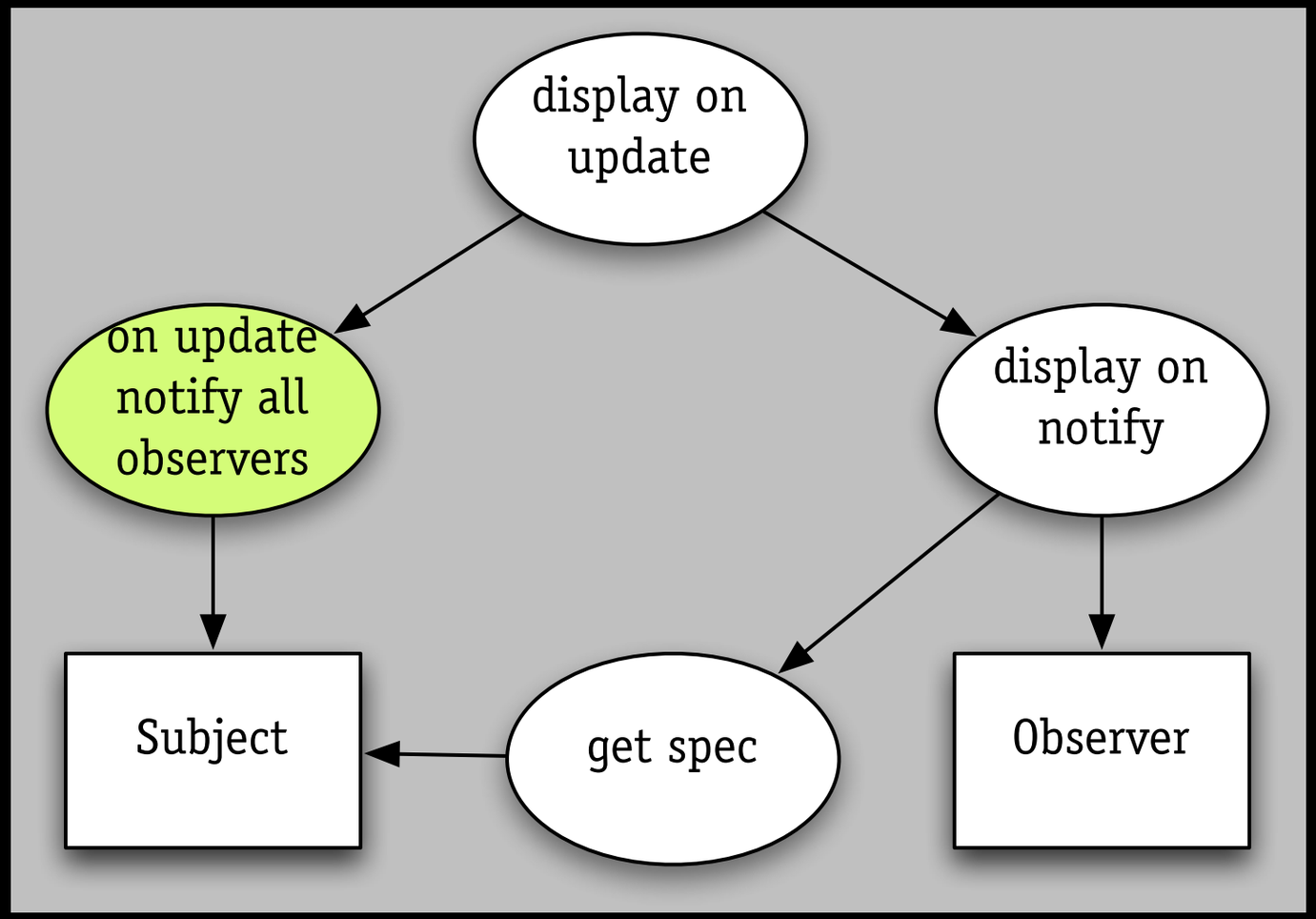


tracing emergency stop



towards automated analysis

observer



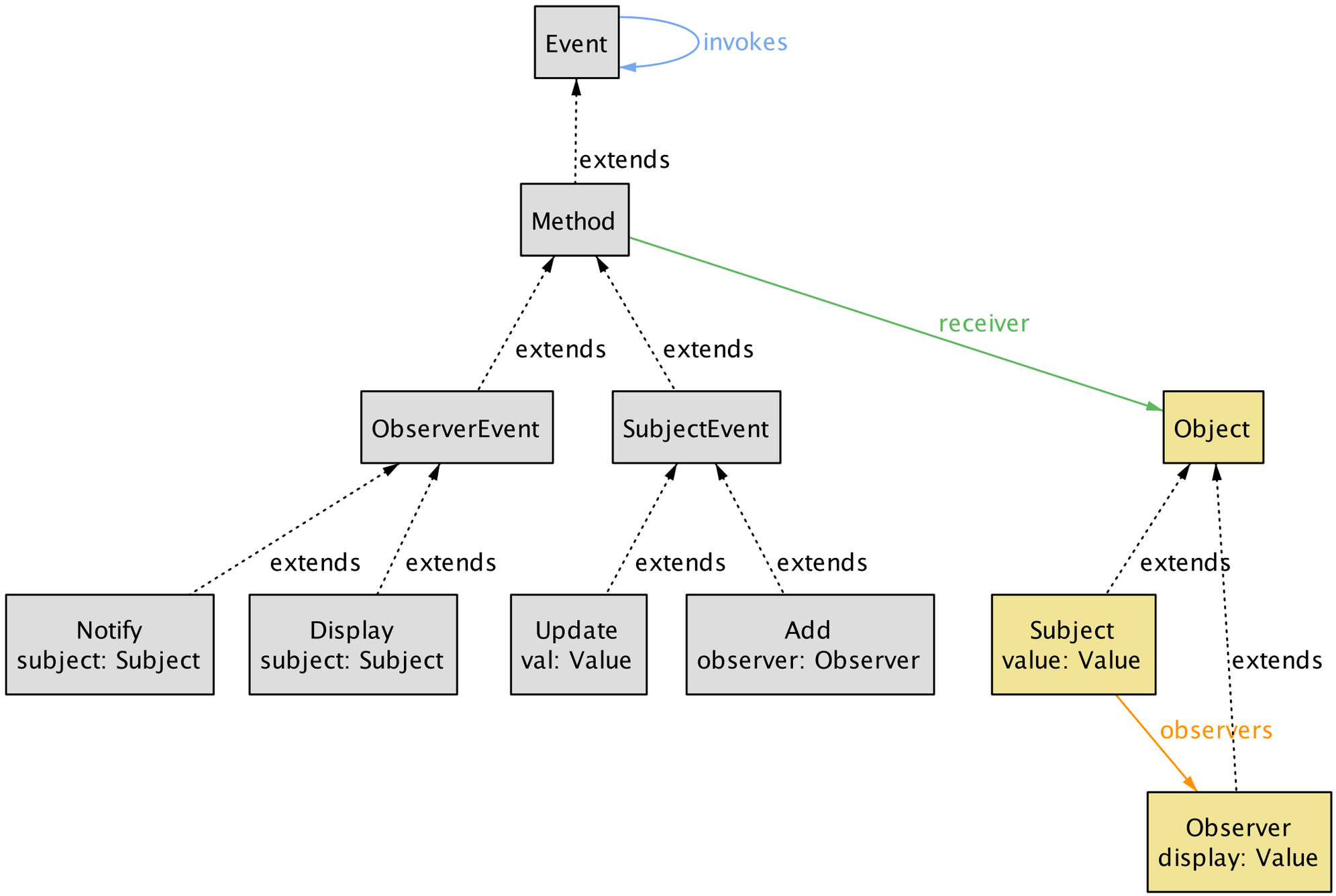
how to formalize?

like this?

```
contract SubjectView
    Subject supports [
        value : Value
        SetValue(val:Value) ↦ Δvalue {value = val}; Notify()
        GetValue() : Value ↦ return value
        Notify() ↦ ⟨|| v : v ∈ Views : v→Update() ⟩

        AttachView(v:View) ↦ {v ∈ Views}
        DetachView(v:View) ↦ {v ∉ Views}
        ]
    Views : Set(View) where each  View supports [
        Update() ↦ Draw()
        Draw() ↦ Subject→GetValue() {View reflects Subject.value}
        SetSubject(s:Subject) ↦ {Subject = s}
        ]
    invariant
        Subject.SetValue(val) ↦ ⟨∀v : v ∈ Views : v reflects Subject.value⟩
    instantiation
        ⟨|| v : v ∈ Views : ⟨Subject→AttachView(v) || v→SetSubject(Subject)⟩ ⟩
end contract
```

Helm, Holland & Gangopadhyay, 1990

alloy object model



class hierarchy

class hierarchy

```
sig Value {}
sig Time {}
```

class hierarchy

```
sig Value {}
sig Time {}

sig Subject extends Object {
      observers: Observer -> Time,
      value: Value one -> Time
}
```

class hierarchy

```
sig Value {}
sig Time {}

sig Subject extends Object {
      observers: Observer -> Time,
      value: Value one -> Time
      }

sig Observer extends Object {
      display: Value one -> Time
      }
```

event hierarchy

event hierarchy

```
abstract sig Event {
    before, after: Time,
    invokes: set Event
    }
```

event hierarchy

```
abstract sig Event {
    before, after: Time,
    invokes: set Event
    }

abstract sig Method extends Event {
    receiver: Object
    }
```

event hierarchy

```
abstract sig Event {
    before, after: Time,
    invokes: set Event
    }

abstract sig Method extends Event {
    receiver: Object
    }

abstract sig SubjectEvent extends Method {} {
    receiver in Subject
    }
```

event hierarchy

```
abstract sig Event {
      before, after: Time,
      invokes: set Event
      }

abstract sig Method extends Event {
      receiver: Object
      }

abstract sig SubjectEvent extends Method {} {
      receiver in Subject
      }

sig Add extends SubjectEvent {
      observer: Observer } {
         receiver.observers.after =
               receiver.observers.before + observer
      }
```

invocation

invocation

```
pred control (invokes: Event -> Event) {
    all u: Update |
        all o: u.receiver.observers.(u.before) |
            some n: u.invokes & Notify |
                n.subject = u.receiver && n.receiver = o
    all n: Notify |
        some d: n.invokes & Display |
            d.receiver = n.receiver && d.subject = n.subject
}
```

invocation

```
pred control (invokes: Event -> Event) {
    all u: Update |
        all o: u.receiver.observers.(u.before) |
            some n: u.invokes & Notify |
                n.subject = u.receiver && n.receiver = o
    all n: Notify |
        some d: n.invokes & Display |
            d.receiver = n.receiver && d.subject = n.subject
    }

fact MinimalInvocation {
    control [invokes]
    no e, e': Event | e->e' in invokes and control [invokes - e->e']
    }
```

invocation

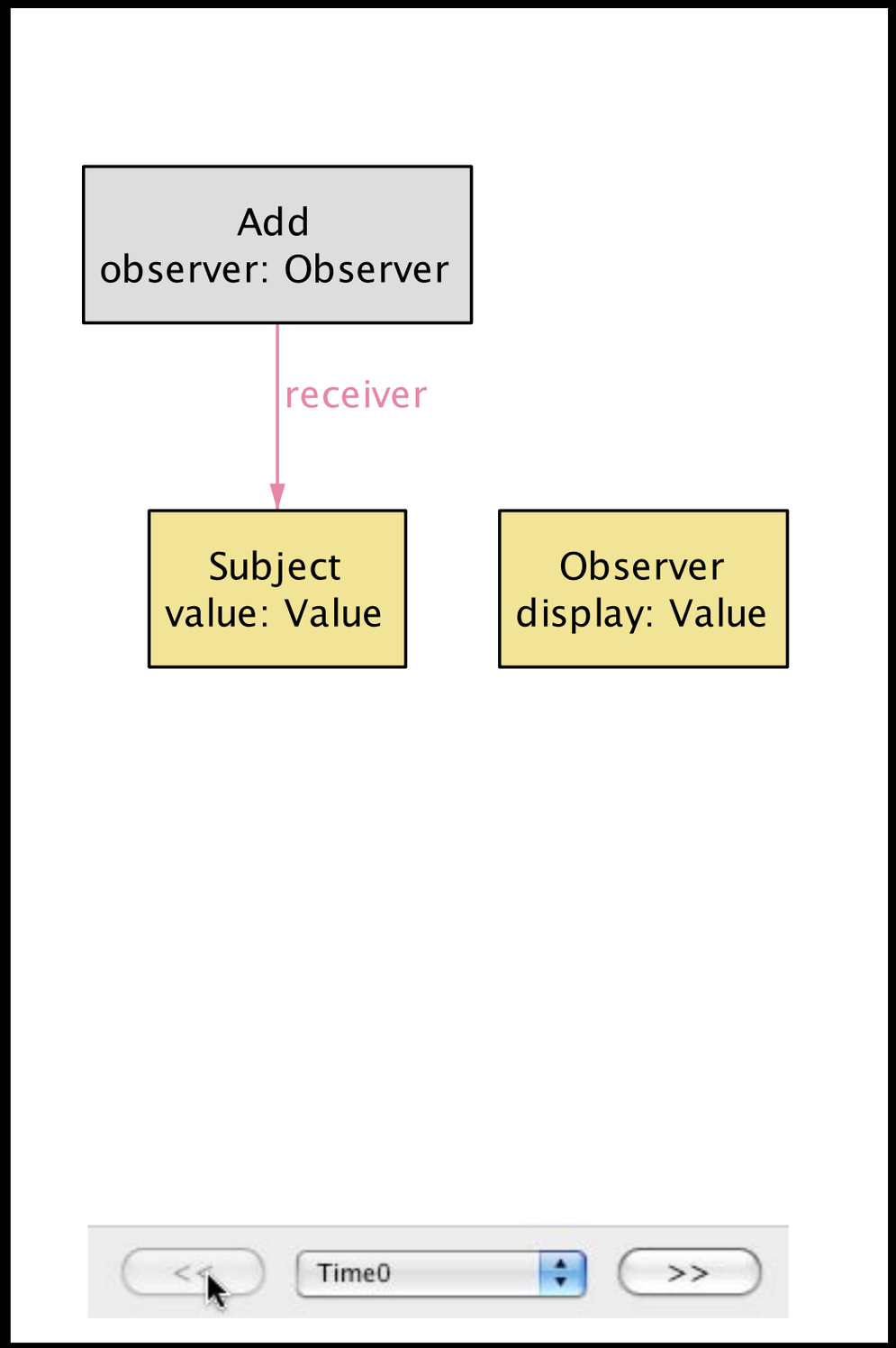
```
pred control (invokes: Event -> Event) {
    all u: Update |
        all o: u.receiver.observers.(u.before) |
            some n: u.invokes & Notify |
                n.subject = u.receiver && n.receiver = o
    all n: Notify |
        some d: n.invokes & Display |
            d.receiver = n.receiver && d.subject = n.subject
    }

fact MinimalInvocation {
    control [invokes]
    no e, e': Event | e->e' in invokes and control [invokes - e->e']
    }
```

invocation

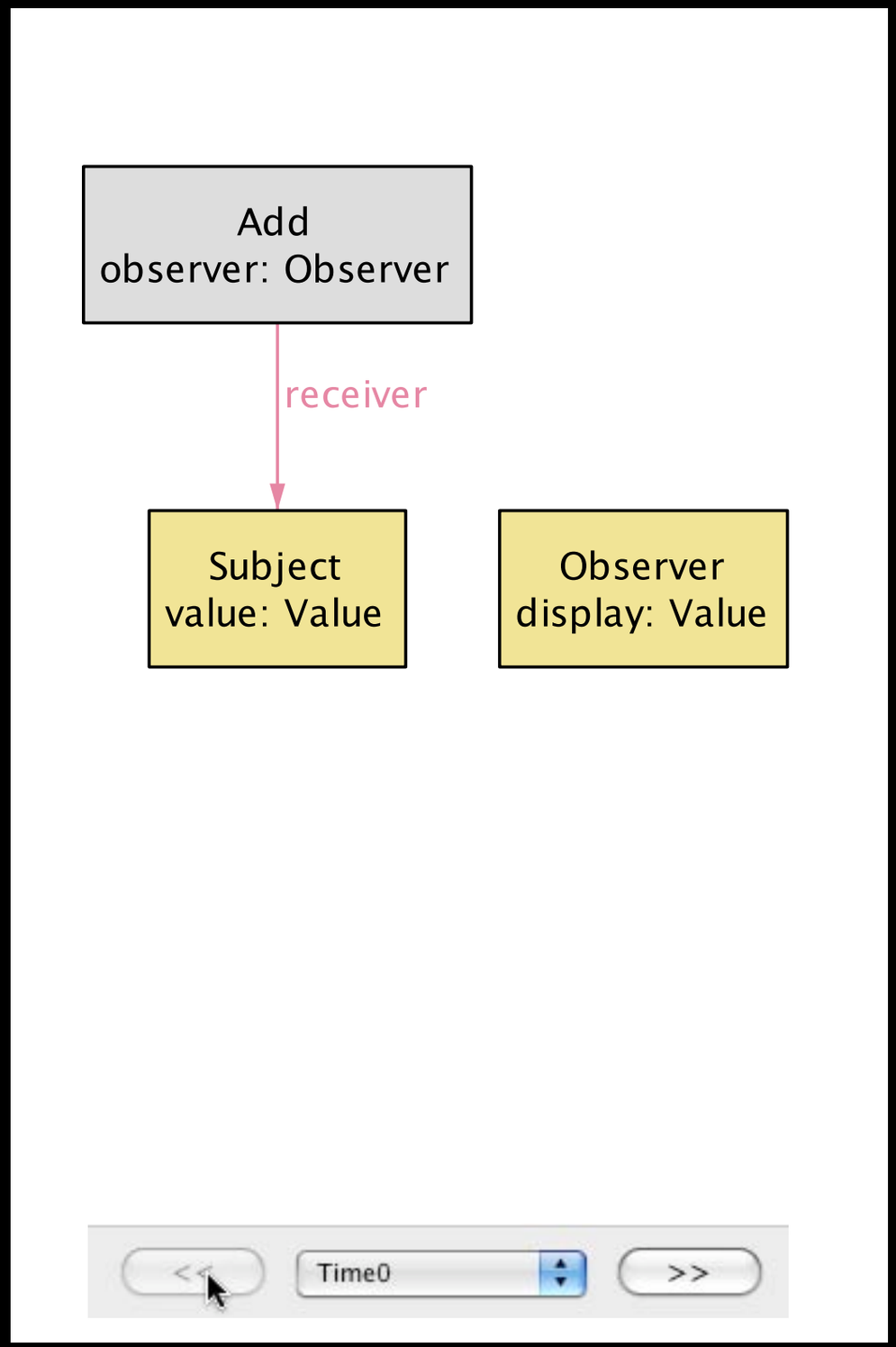
```
pred control (invokes: Event -> Event) {
    all u: Update |
        all o: u.receiver.observers.(u.before) |
            some n: u.invokes & Notify |
                n.subject = u.receiver && n.receiver = o
    all n: Notify |
        some d: n.invokes & Display |
            d.receiver = n.receiver && d.subject = n.subject
    }

fact MinimalInvocation {
    control [invokes]
    no e, e': Event | e->e' in invokes and control [invokes - e->e']
    }
```

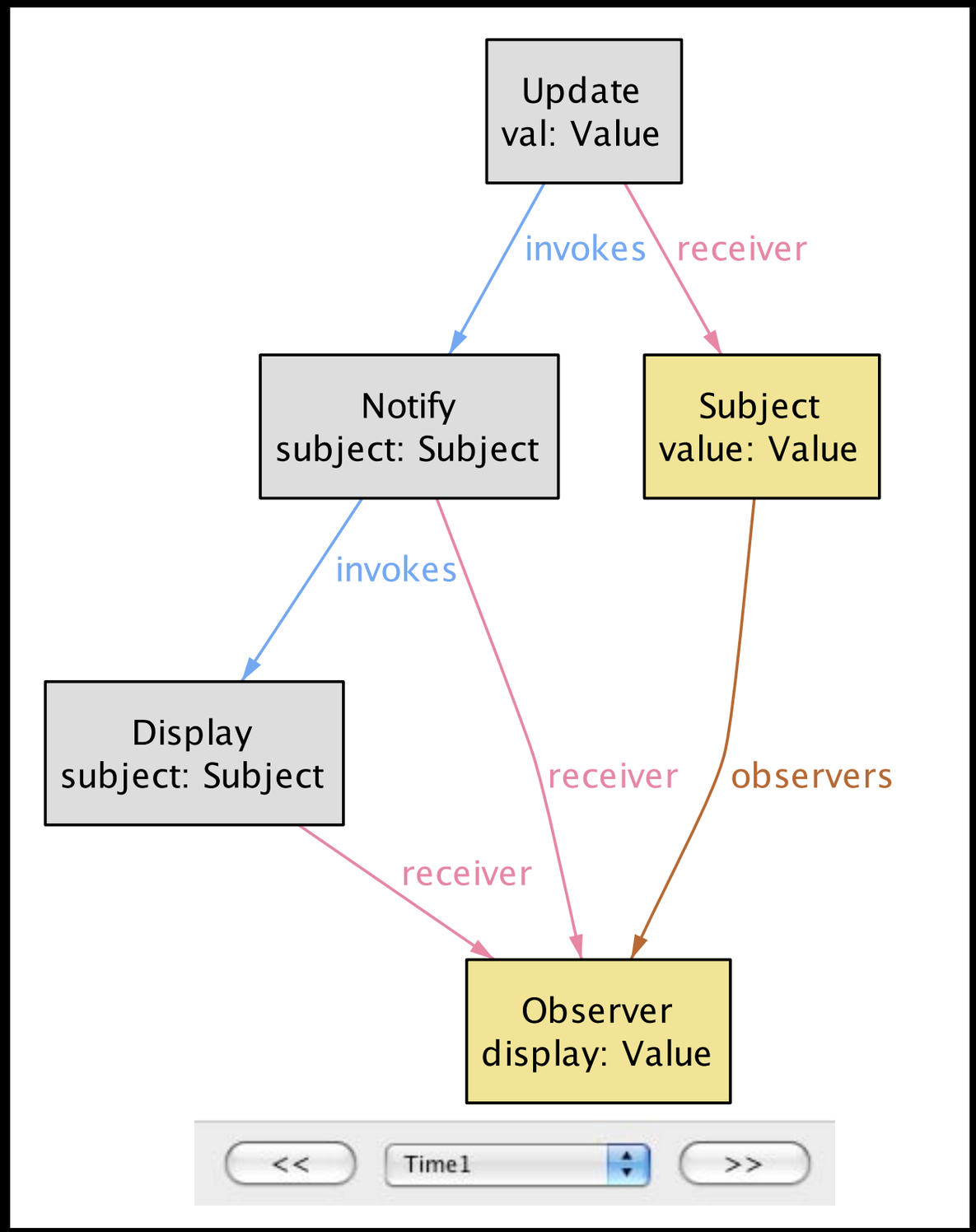
a sample execution

a sample execution



a sample execution





code analysis challenges

where are the properties written?

eg, update results in display

not a property of Subject!

how are properties extracted?

eg, update results in calls to notify

not a change of state

not a data-independent state machine

related work

axiomatic design Suh, 2001

› spec/design *parameters*

design structure matrix Steward; Eppinger; Baldwin/Clark; Lattix

› topological sort of *uses*

evolvability analysis Sullivan et al

› derive DSM from *constraints on parameters*

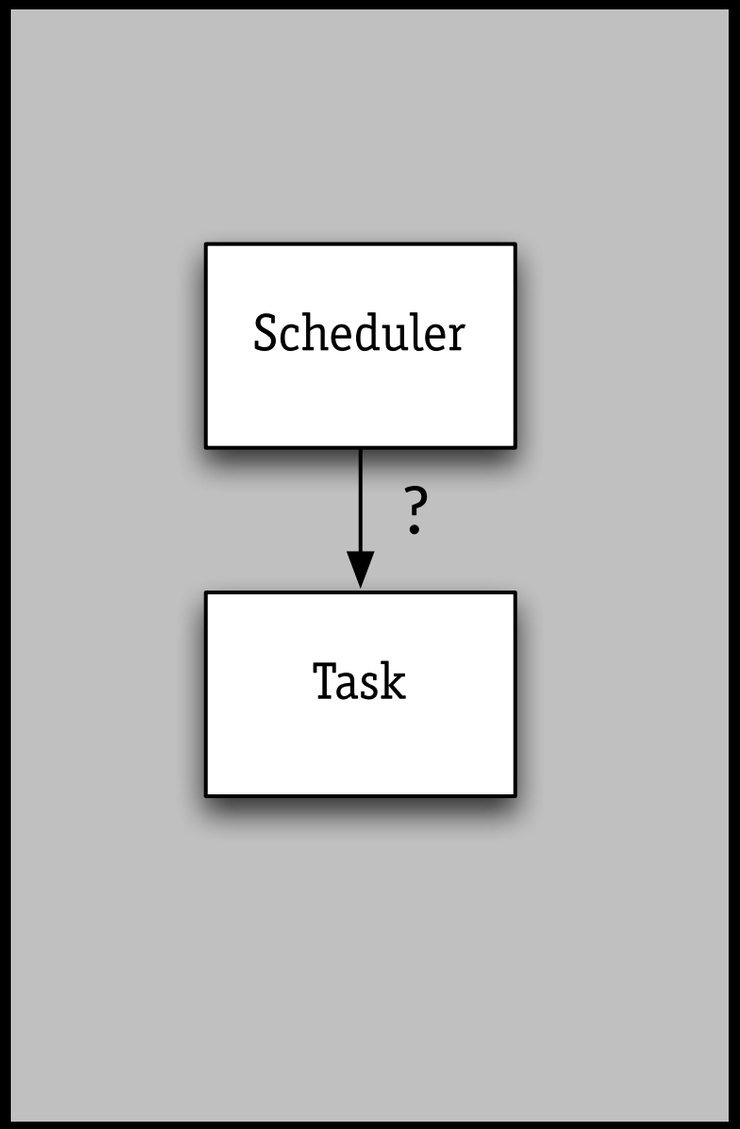
behavioral compositions Helm et al; Barnett, Schulte et al

› operational specs, runtime checks only?

extra slides

uses puzzles

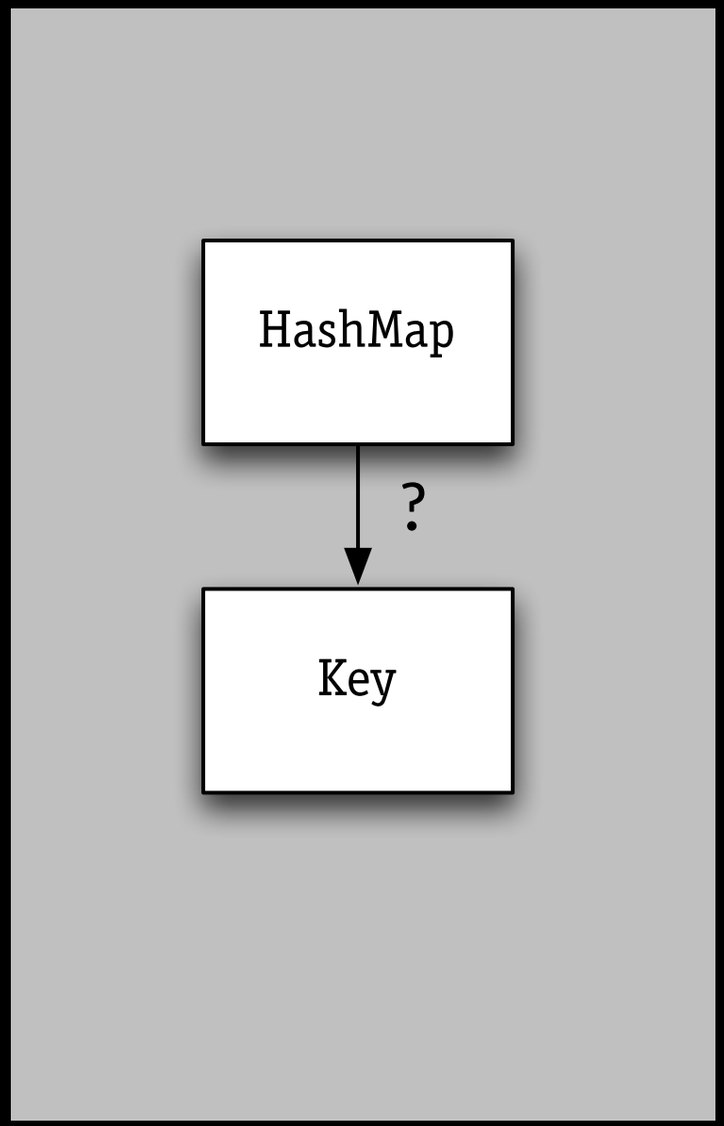
who's using whom?



Scheduler calls Task

but who serves whom?

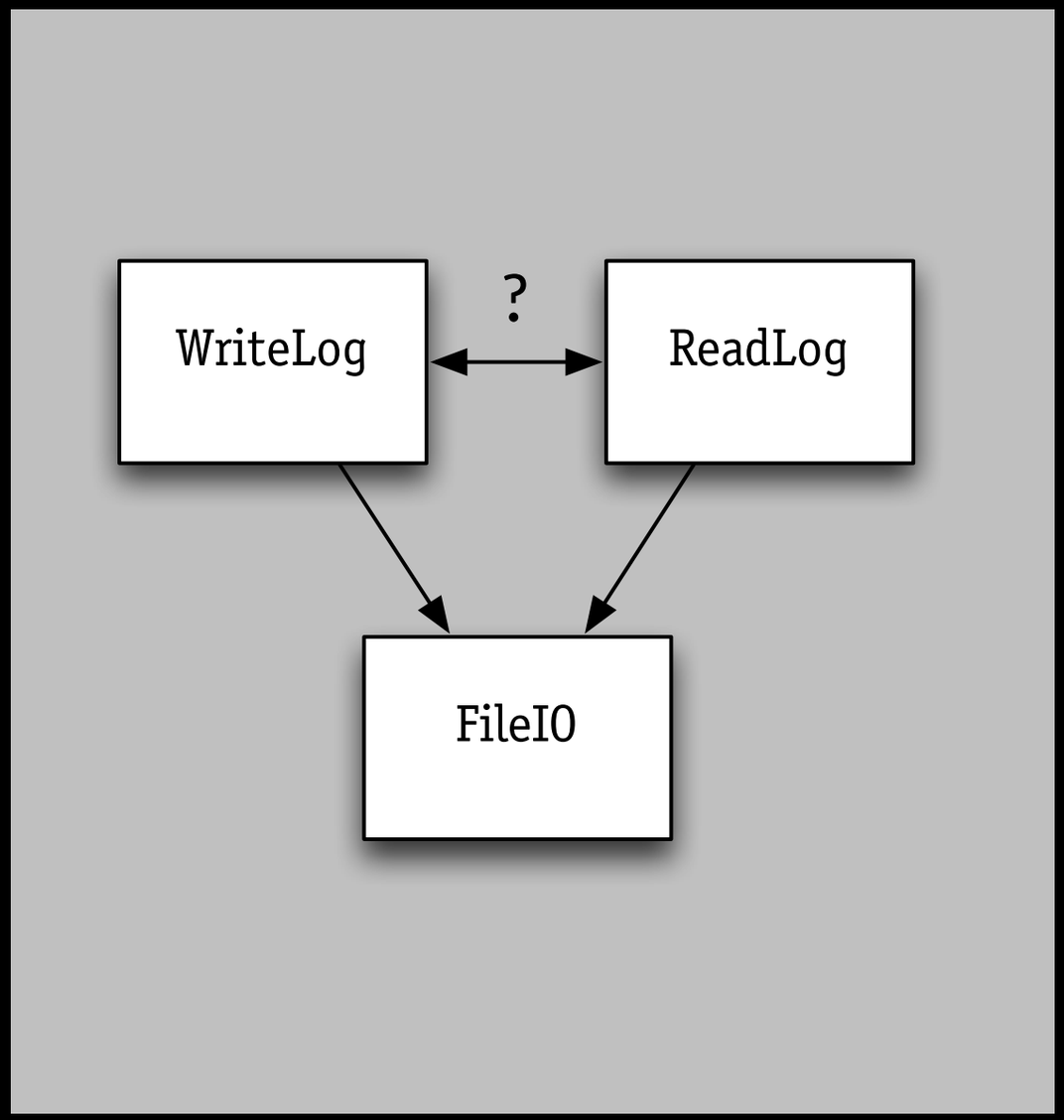
using but not knowing?



HashMap calls Key's equals and hashCode methods

But Key didn't even exist when HashMap was written and tested!

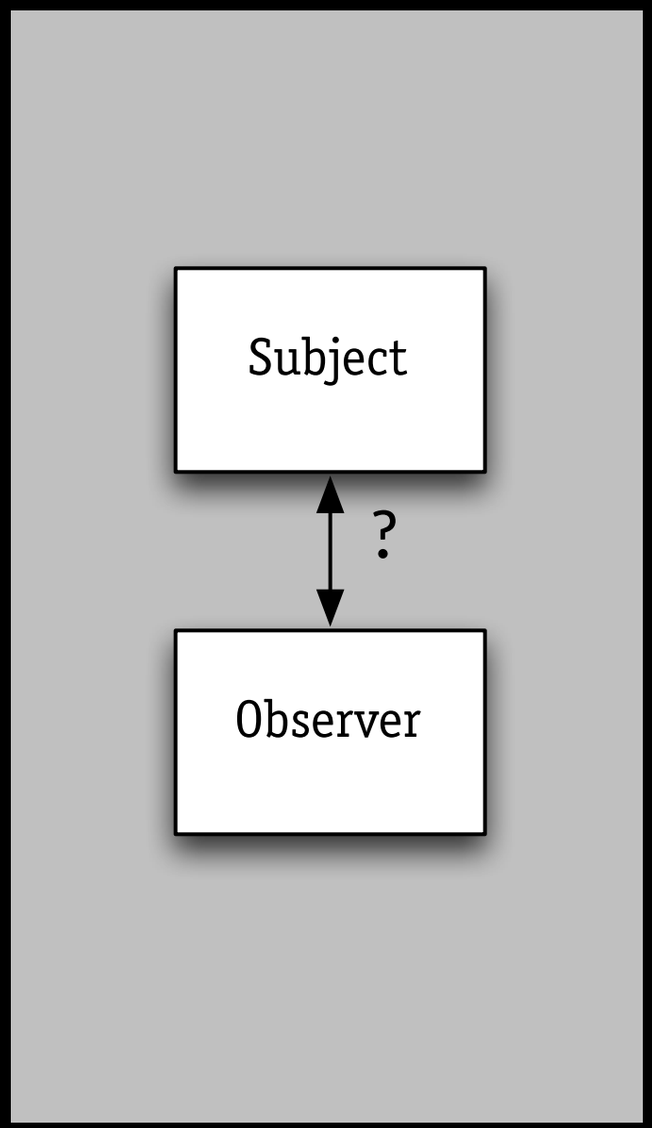
coupling on format?



ReadLog relies on WriteLog for correct formatting.

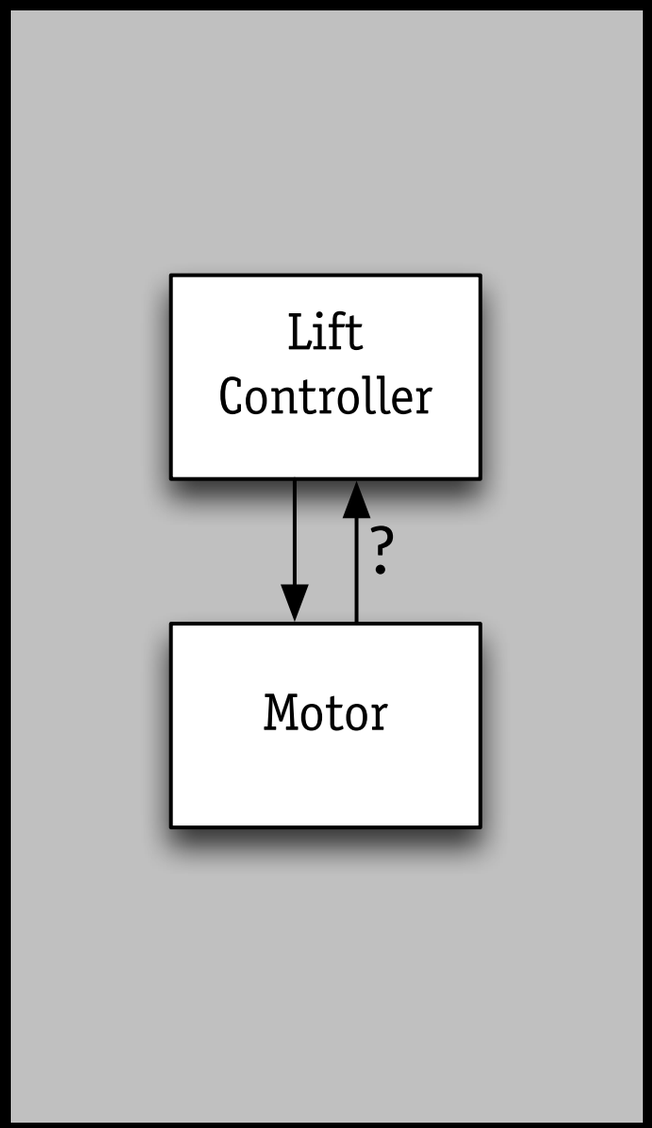
So does it use it?

cyclic uses?



Subject calls Observer to refresh display, and Observer calls back to Subject to get state.

preconditions?

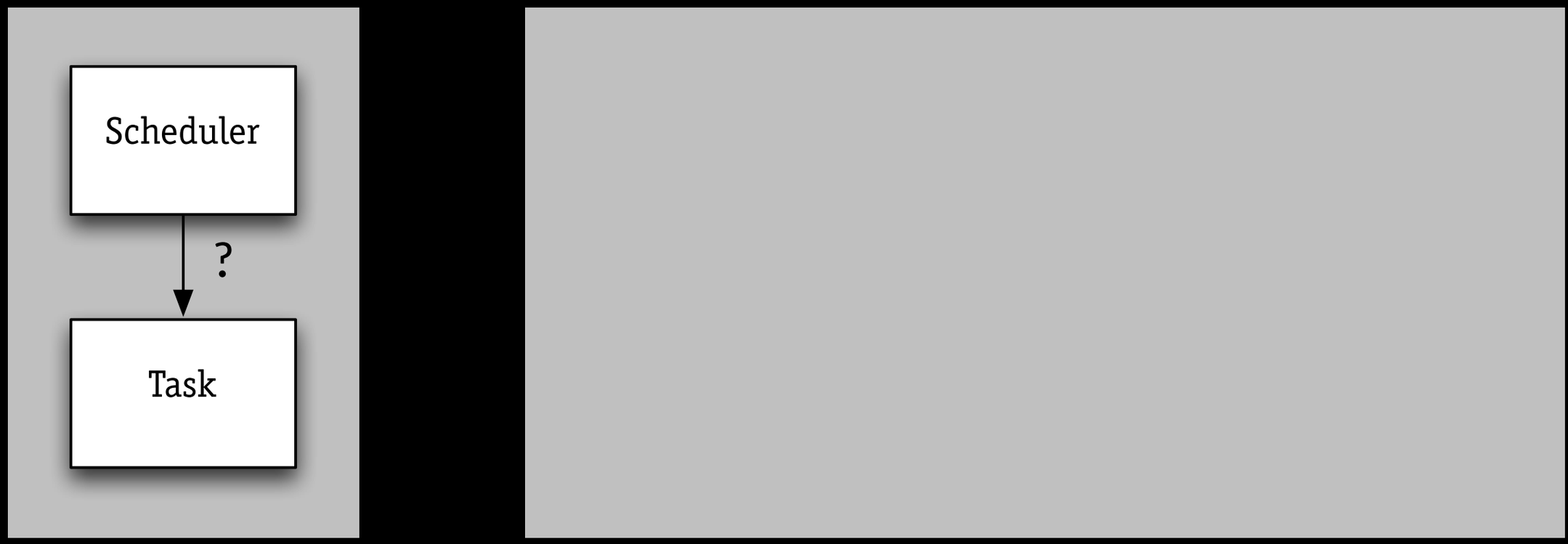


If Motor's precondition is violated, it burns out

So does Motor use Controller too?

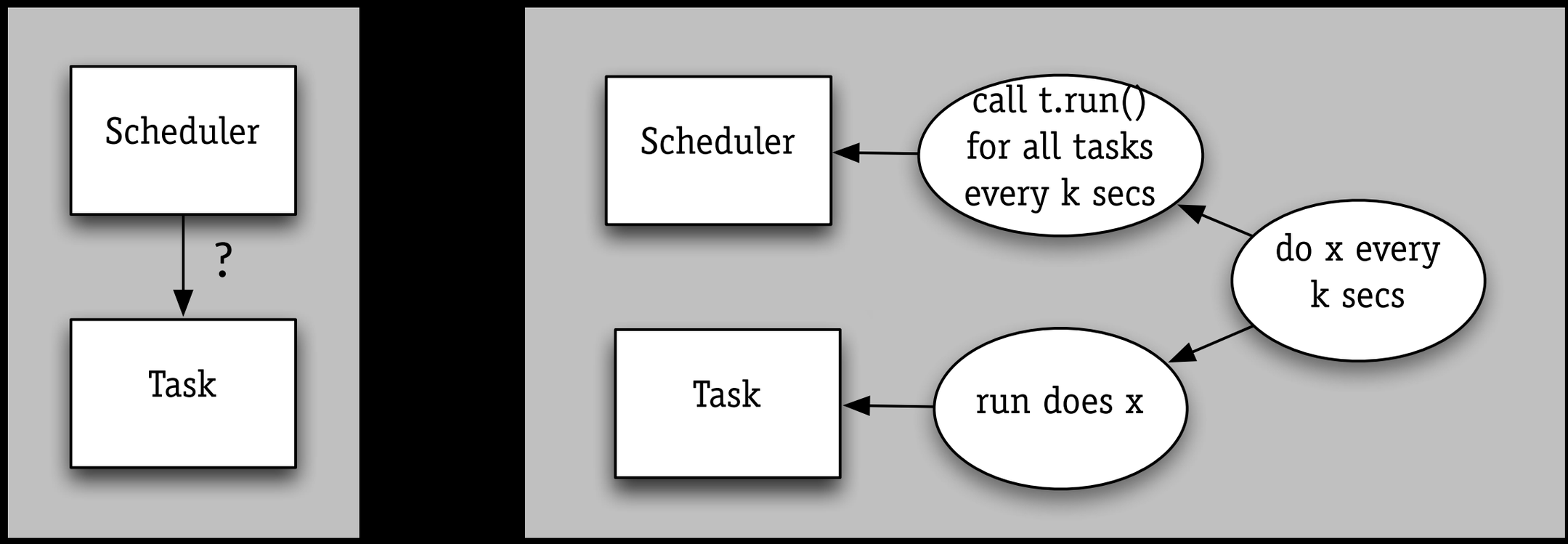
uses puzzles, revisited

who's using whom?



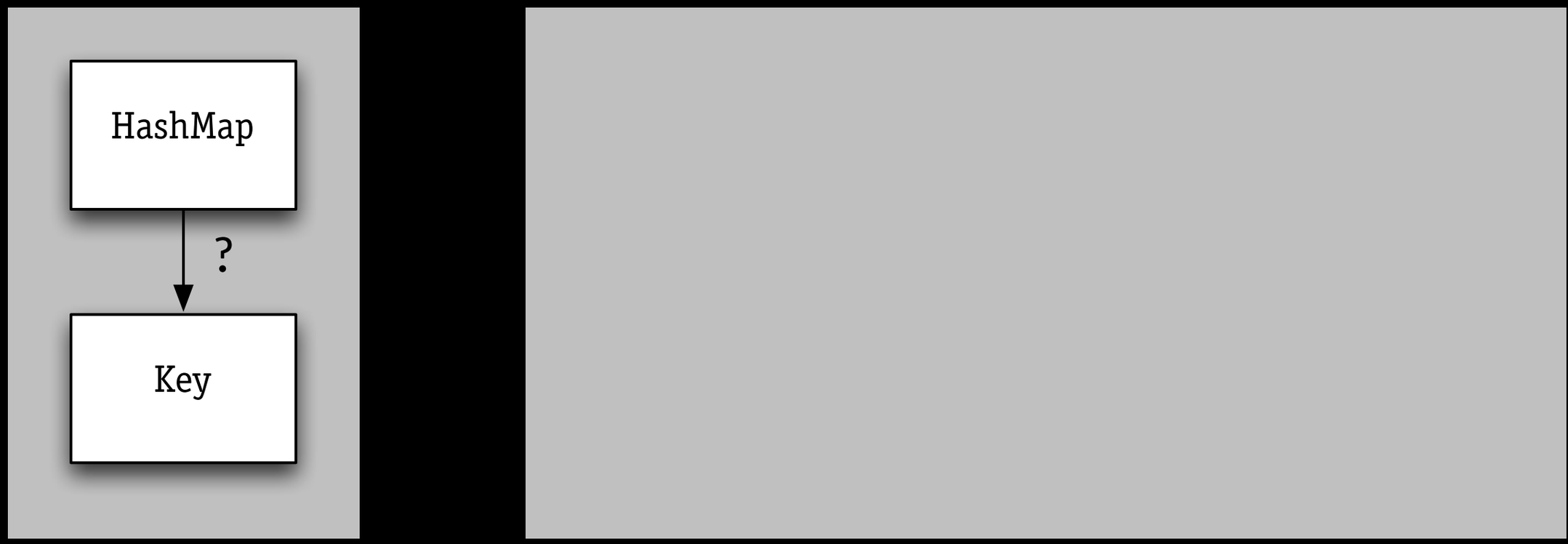
uses doesn't capture coupling by cooperation

who's using whom?



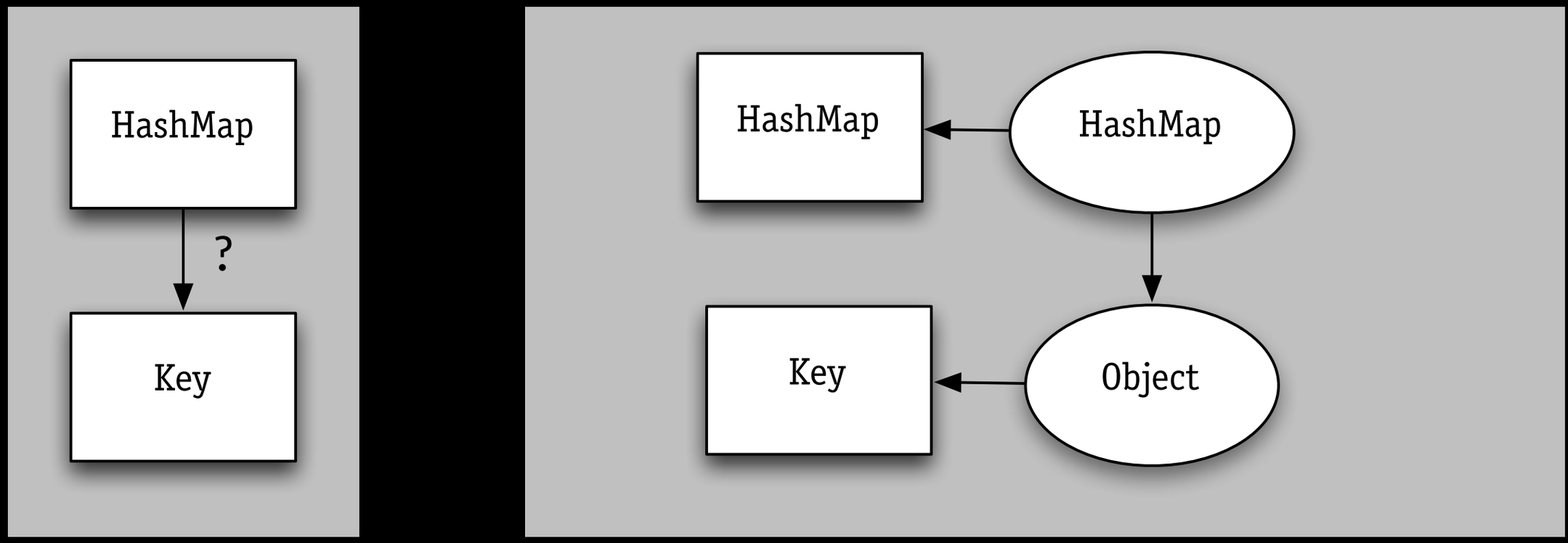
uses doesn't capture coupling by cooperation

using but not knowing?



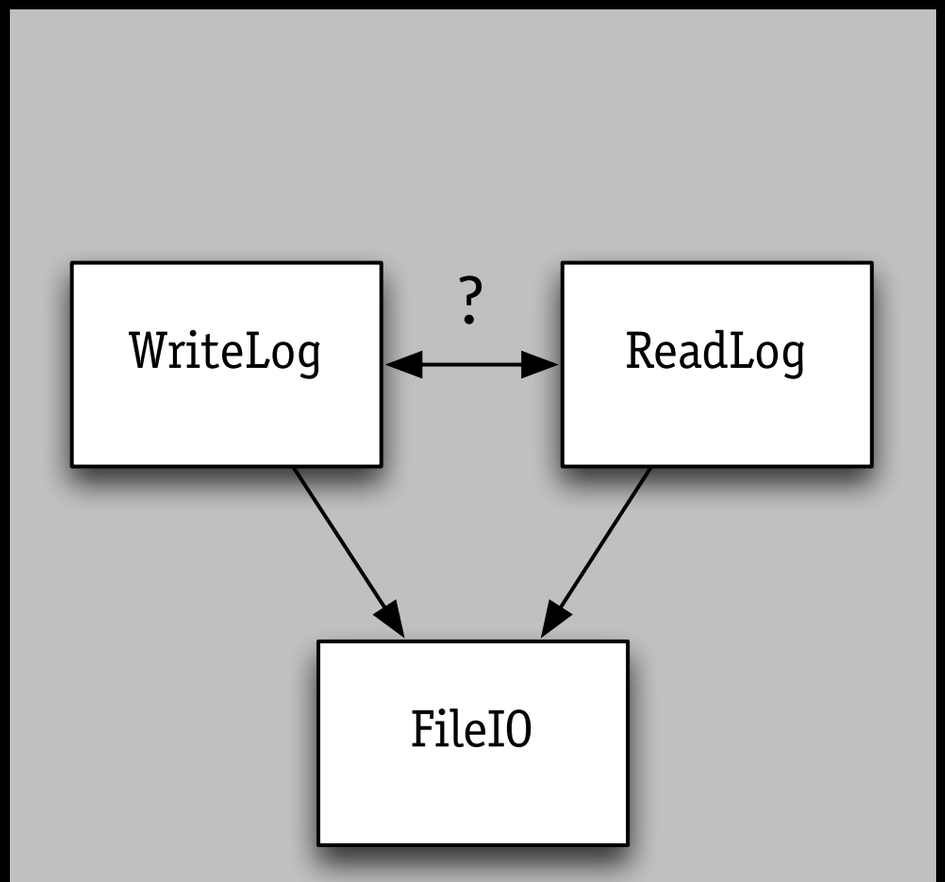
uses is whole spec, not property by property

using but not knowing?



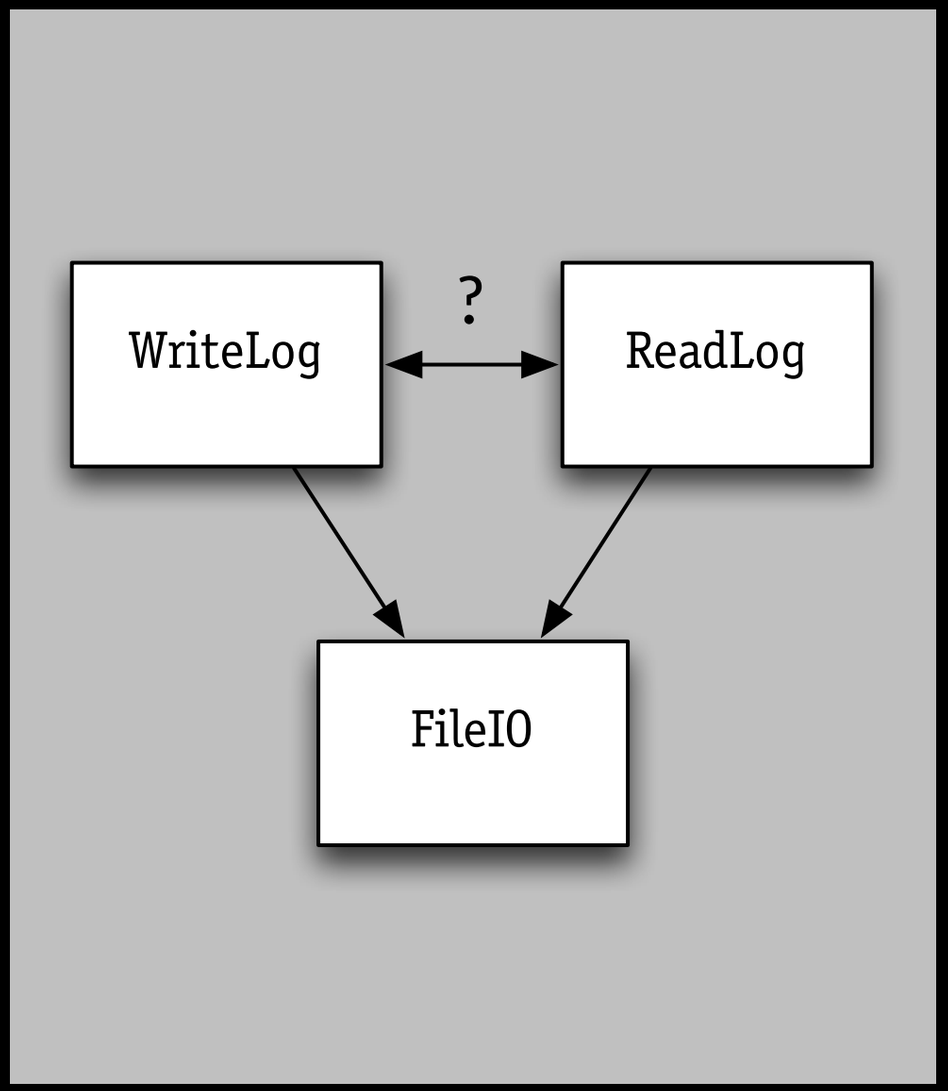
uses is whole spec, not property by property

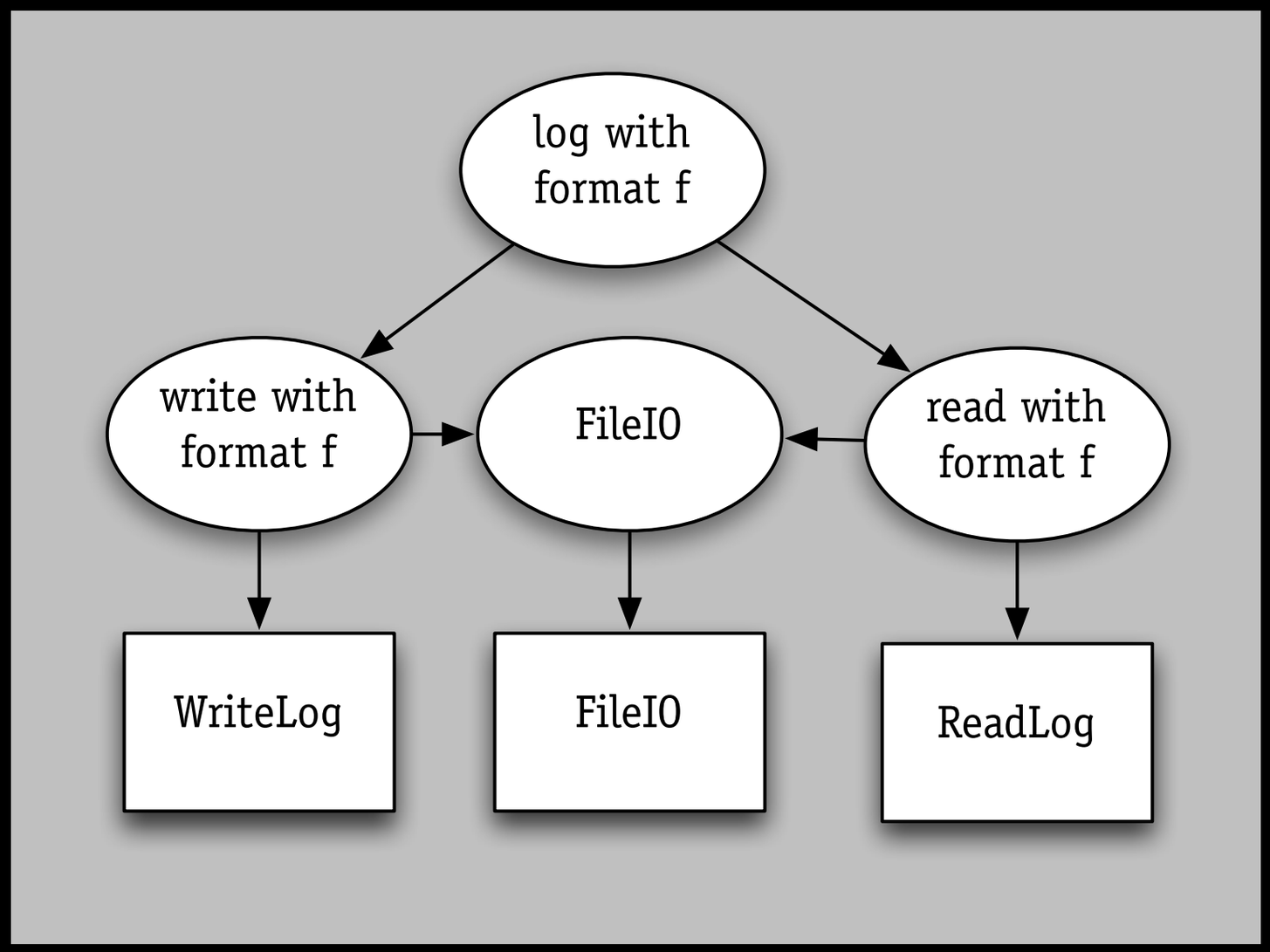
coupling on format?



uses seems to favor the component executing later

coupling on format?



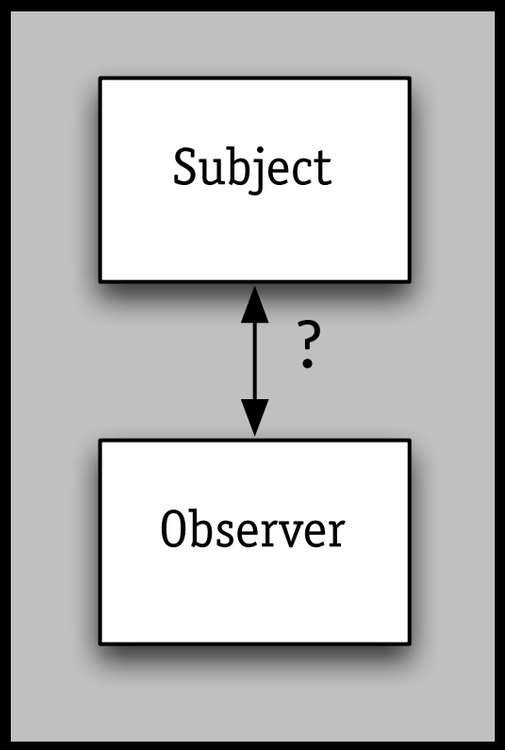


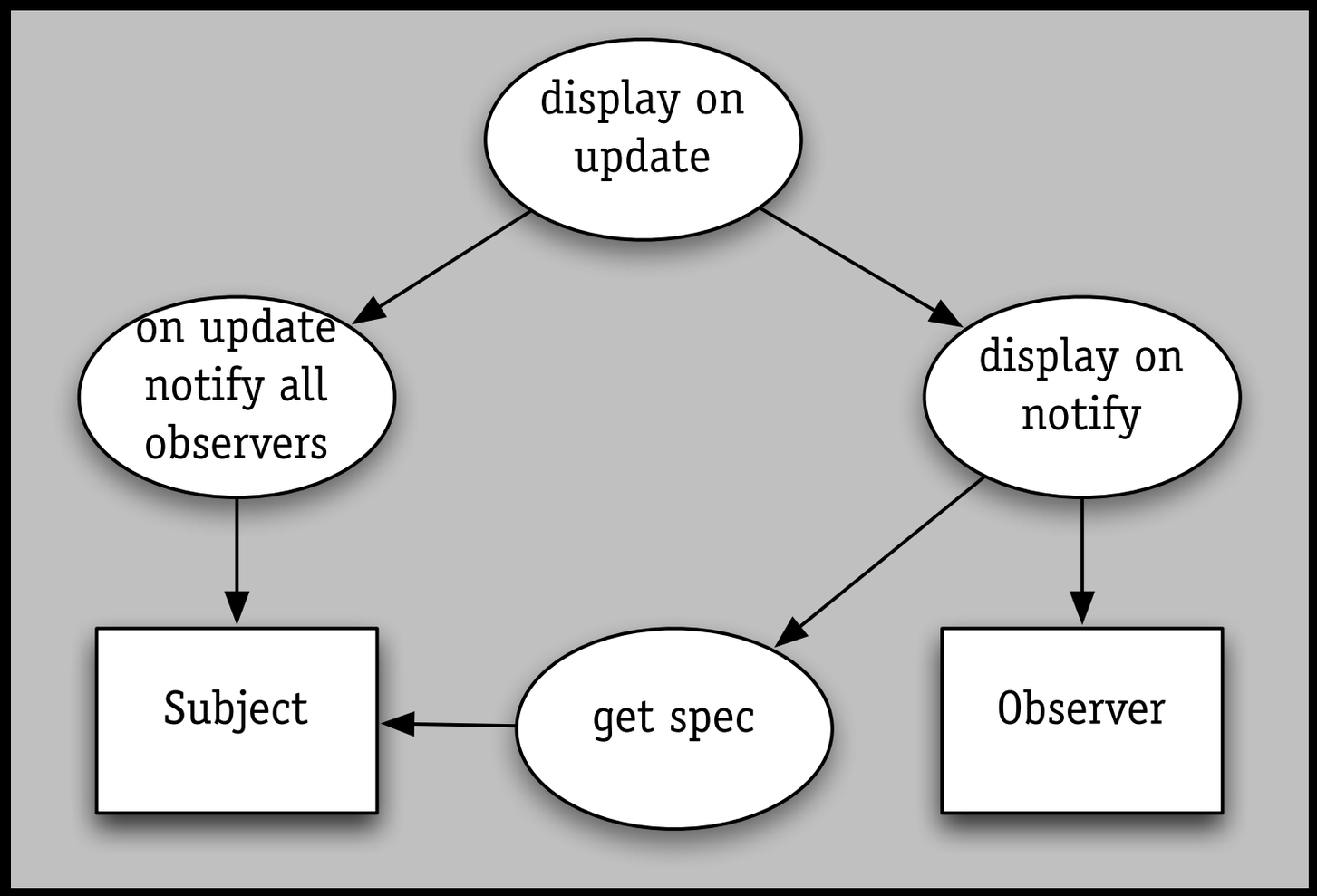
uses seems to favor the component executing later

cyclic uses?

uses works better for classic ADTs than OOPs

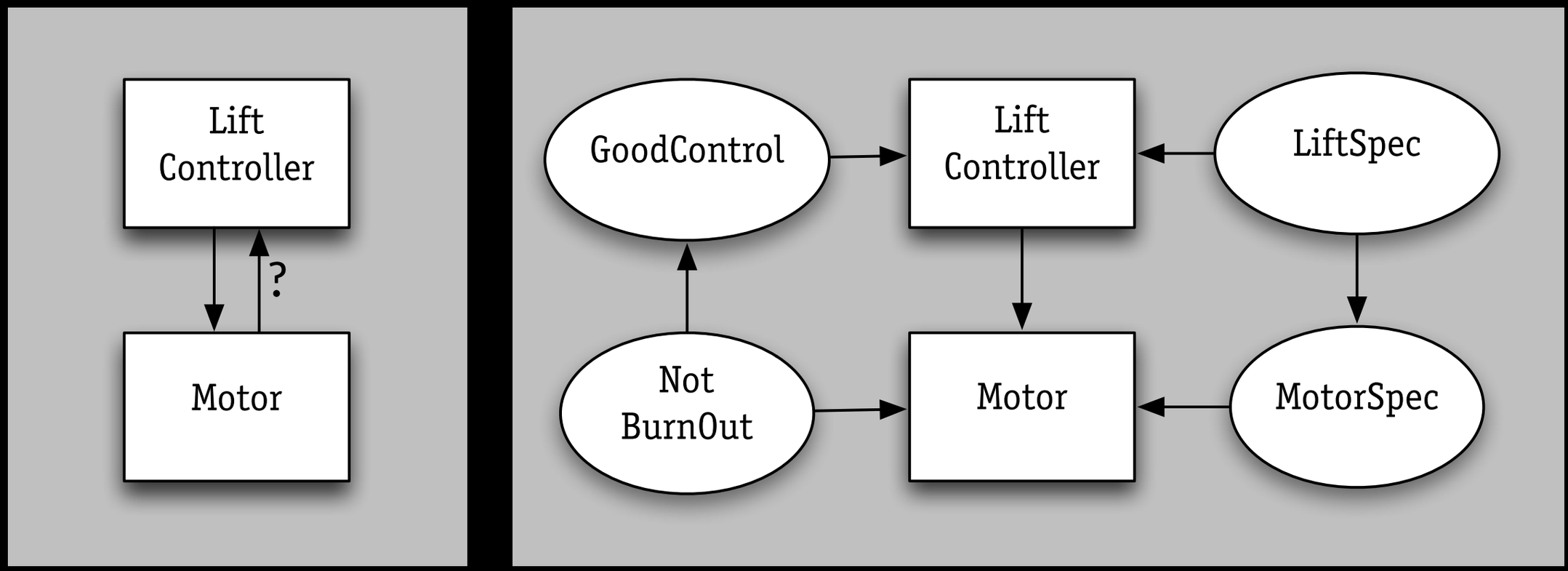
cyclic uses?





uses works better for classic ADTs than OOPs

preconditions?



in uses, component breakage matters only for user

design secrets

design secrets

challenge
› local definition
› global reasoning

idea
› in reasoning,
  treat as *uninterpreted*

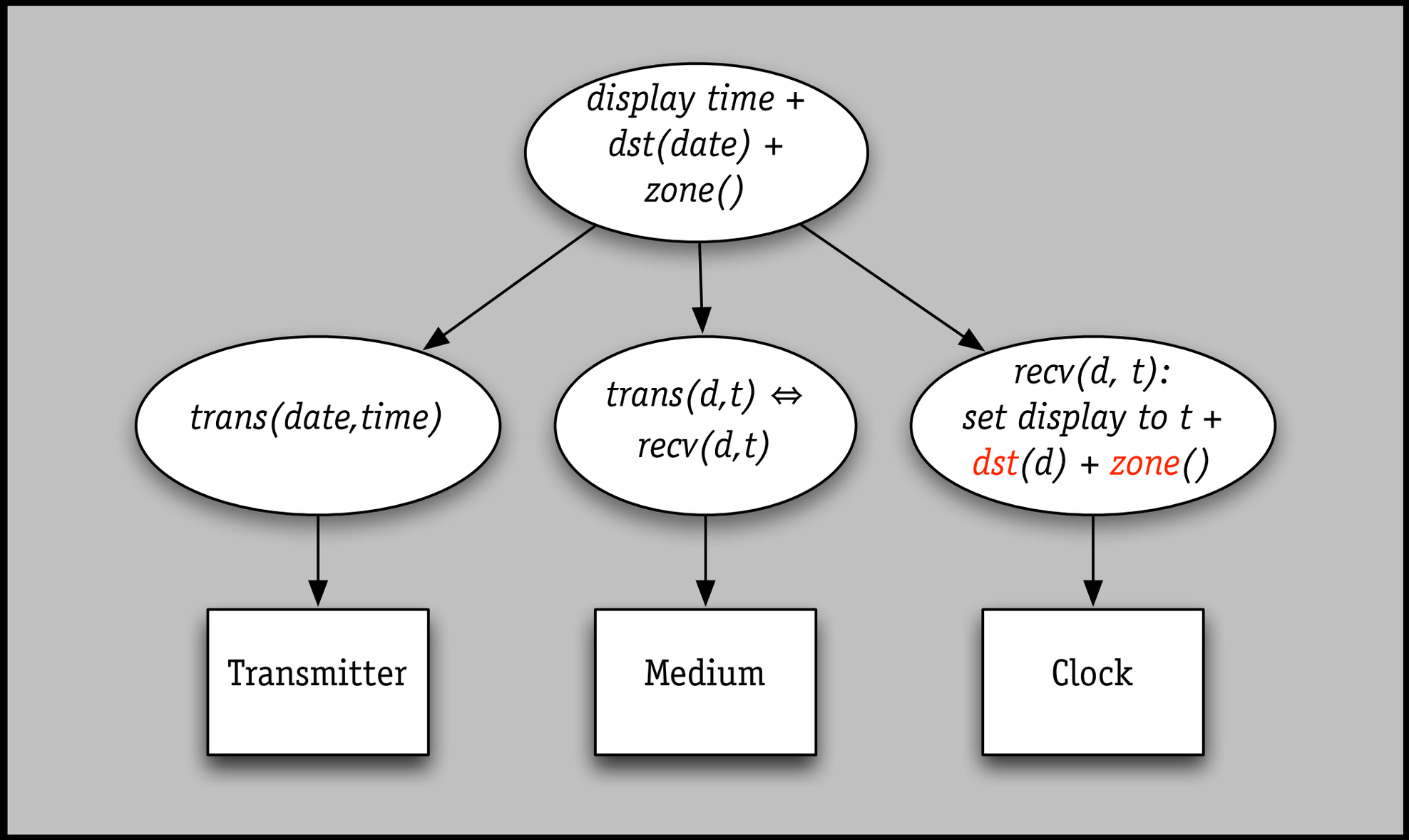
radio-controlled clock

signal from NIST station

handling time zones
› NIST transmits UTC
› switch on clock for zone

handling daylight savings
› transmitted with signal
› ... but ignored by my clock :-(

radio-controlled clock



dst function is local to Clock property