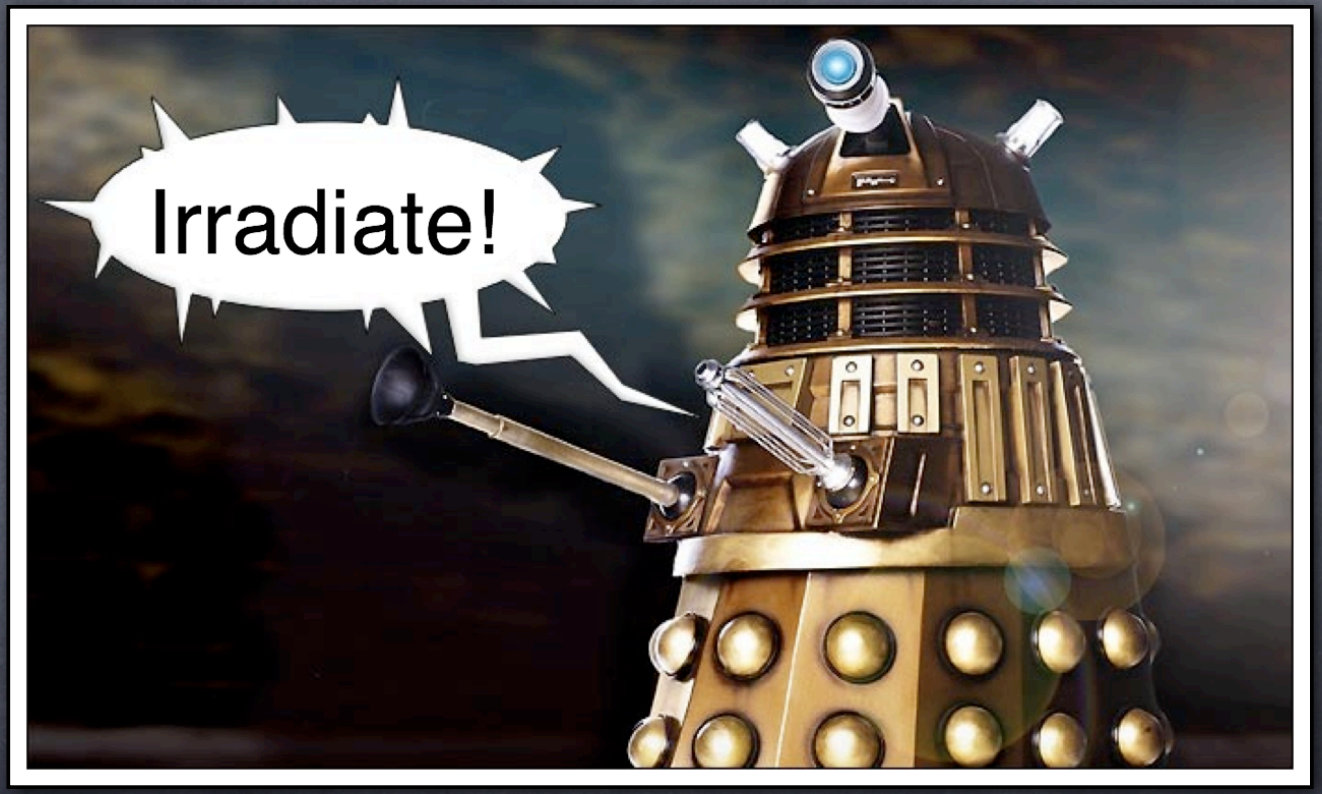
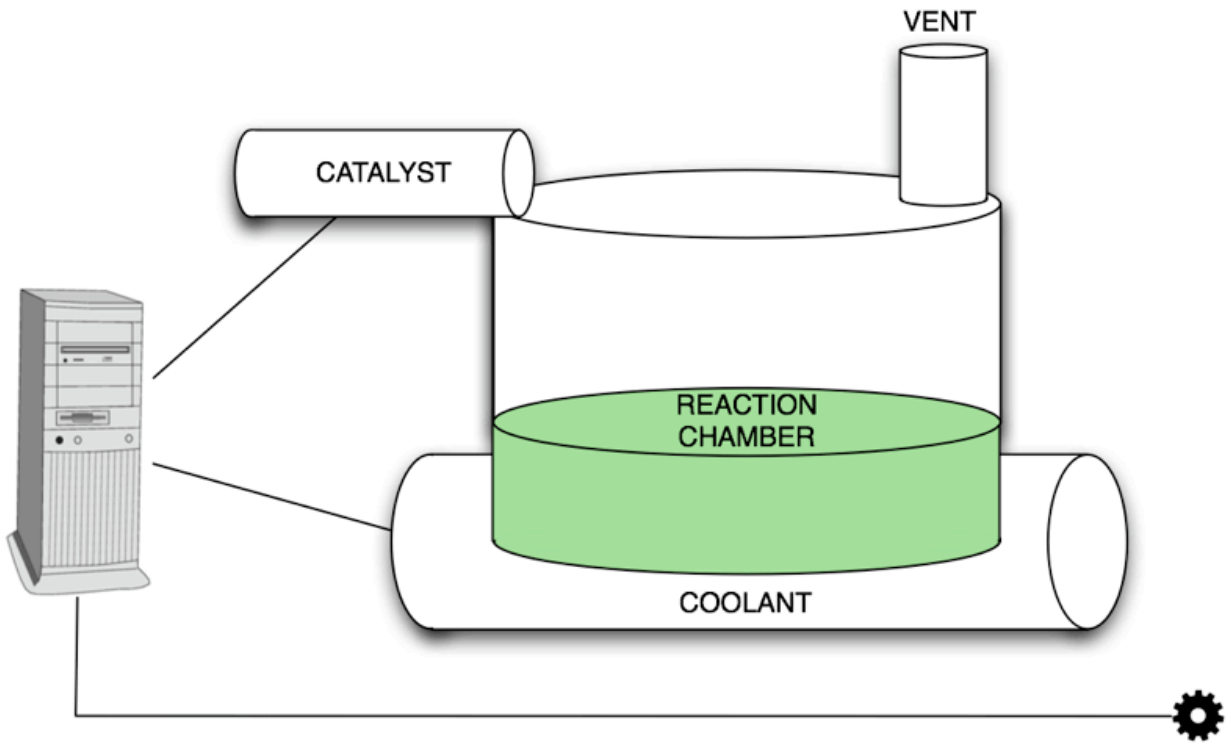
Building Dependability Arguments for Software-Intensive Systems

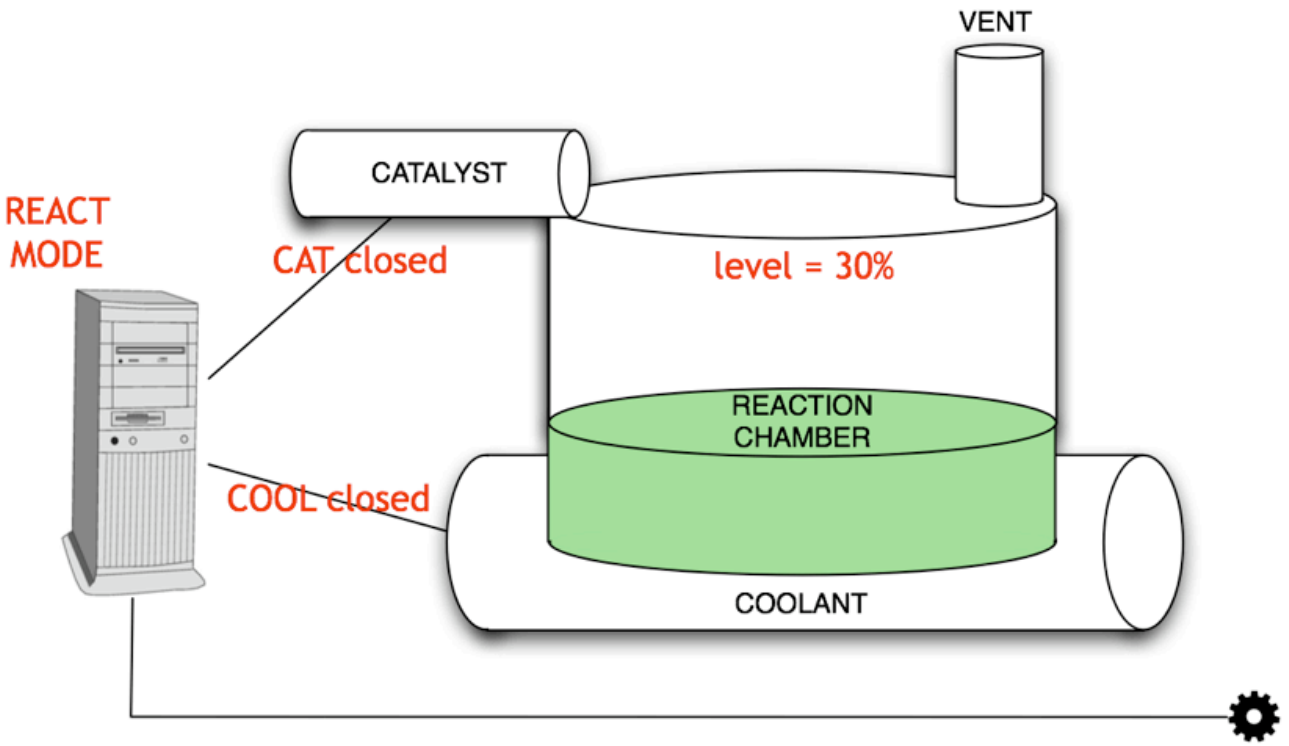


Robert Seater
February 3rd, 2009

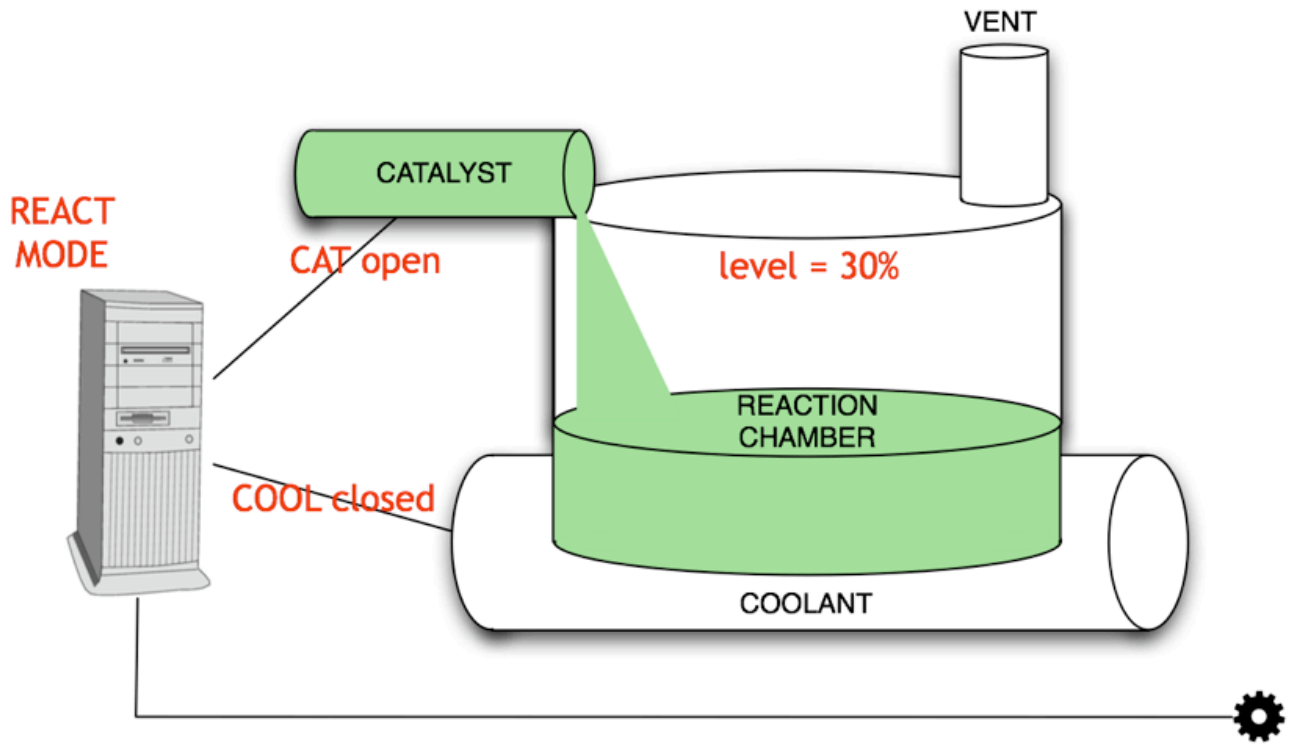
England, 1980



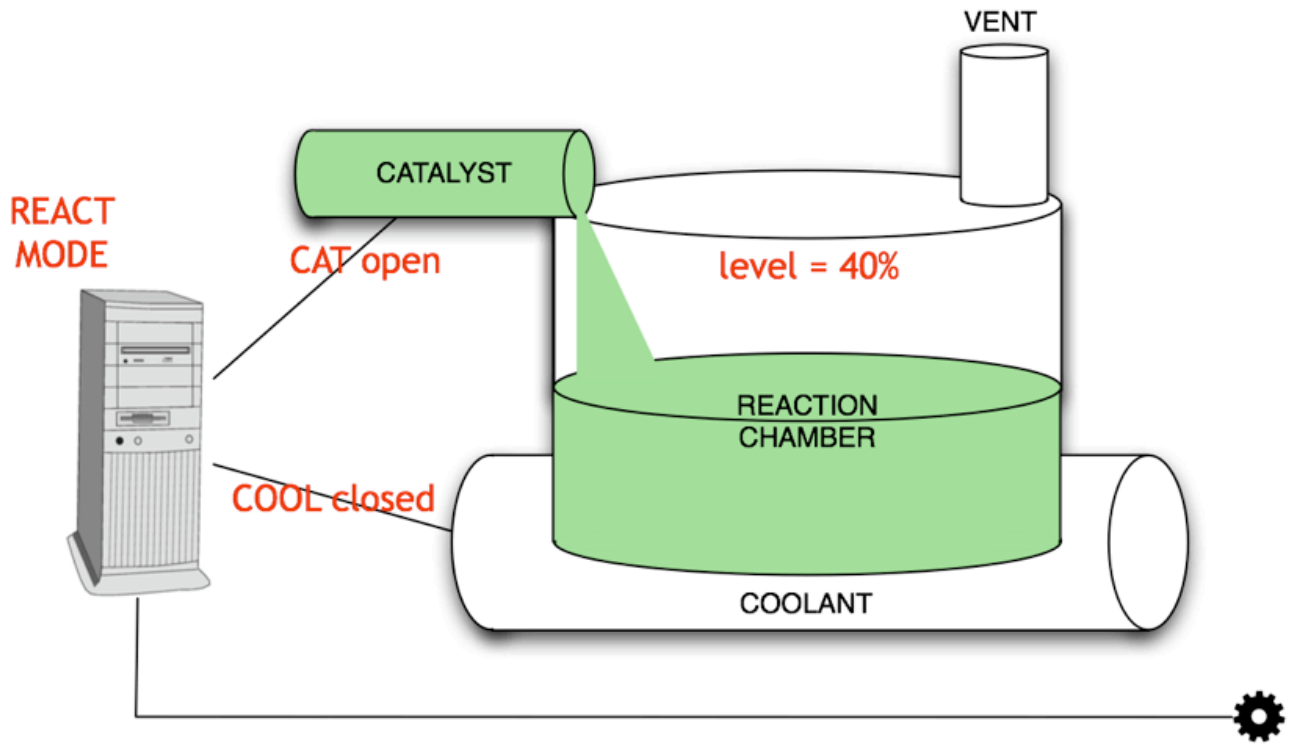
How it works


VENT
CATALYST
REACT
MODE
CAT closed
level = 30%
REACTION
CHAMBER
COOL closed
COOLANT

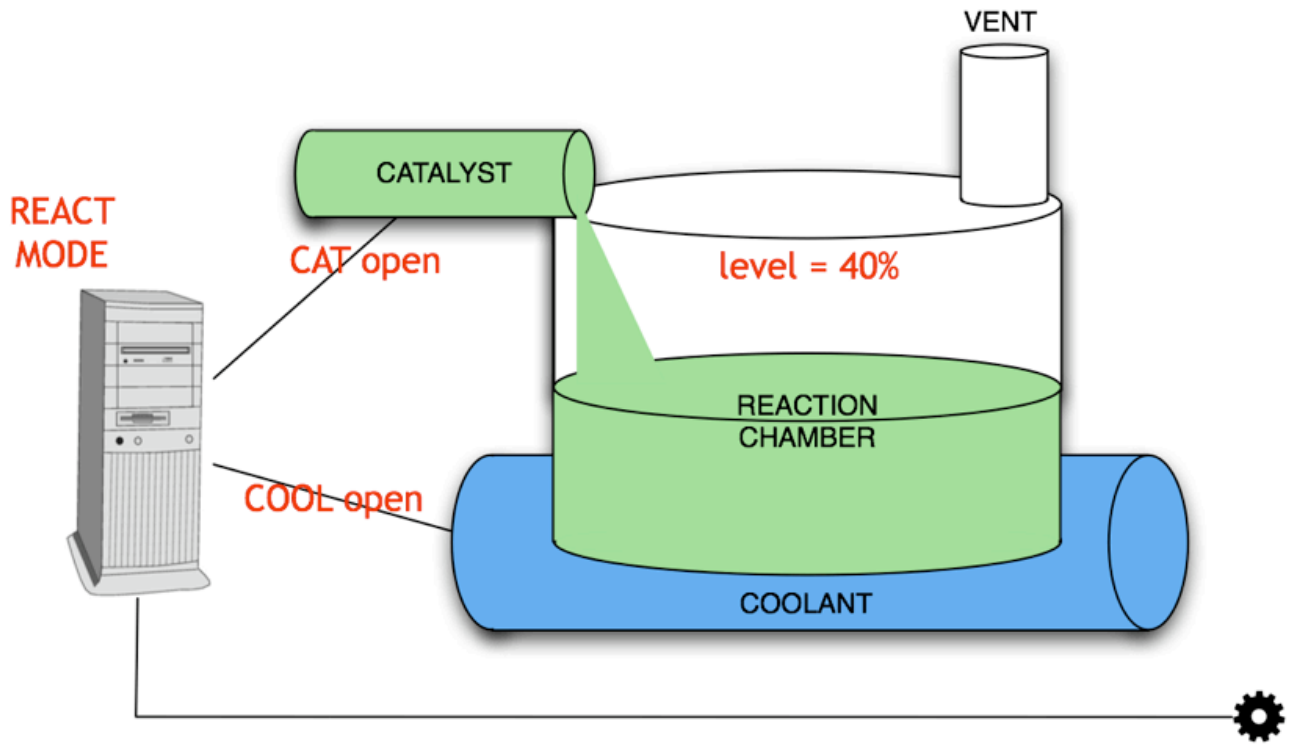
How it works


VENT
CATALYST
REACT
MODE
CAT open
level = 30%
REACTION
CHAMBER
COOL closed
COOLANT

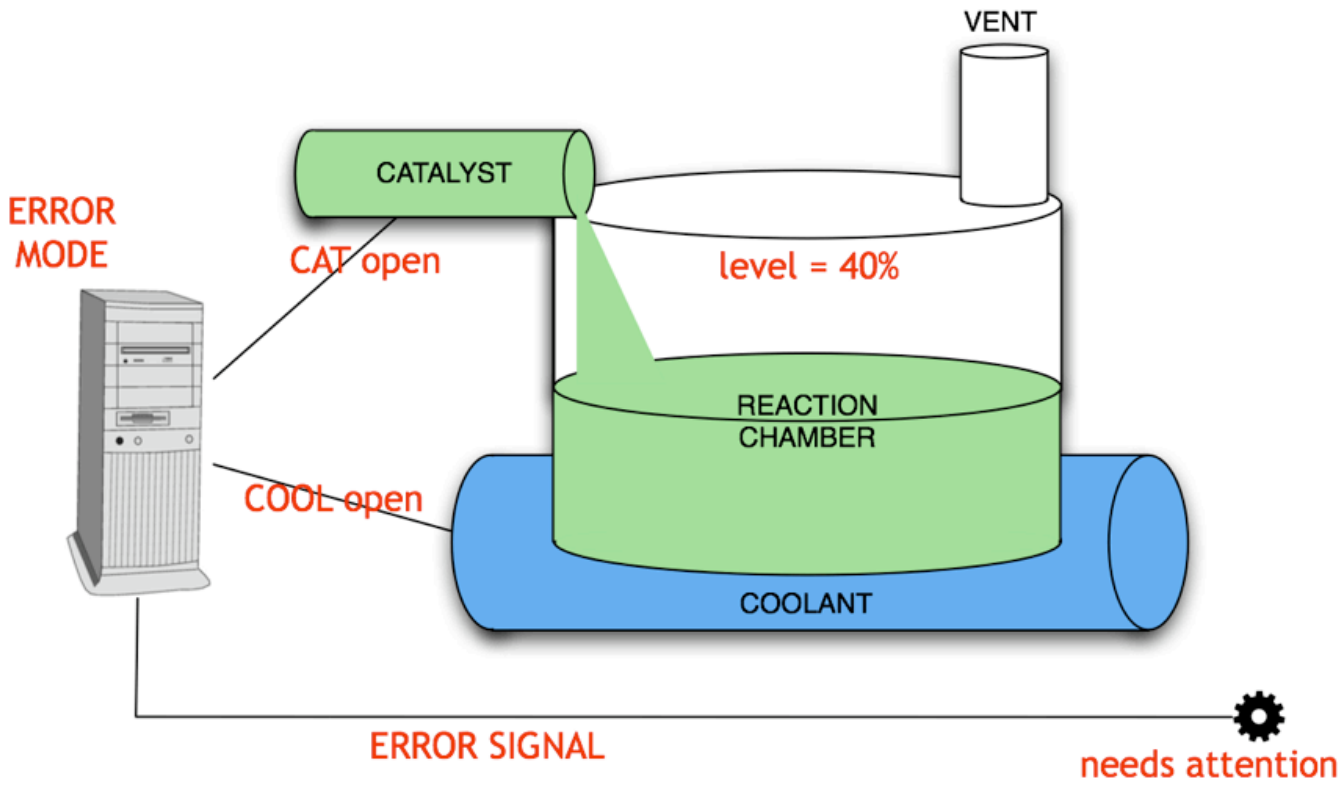
How it works


VENT
CATALYST
REACT
MODE
CAT open
level = 40%
REACTION
CHAMBER
COOL closed
COOLANT

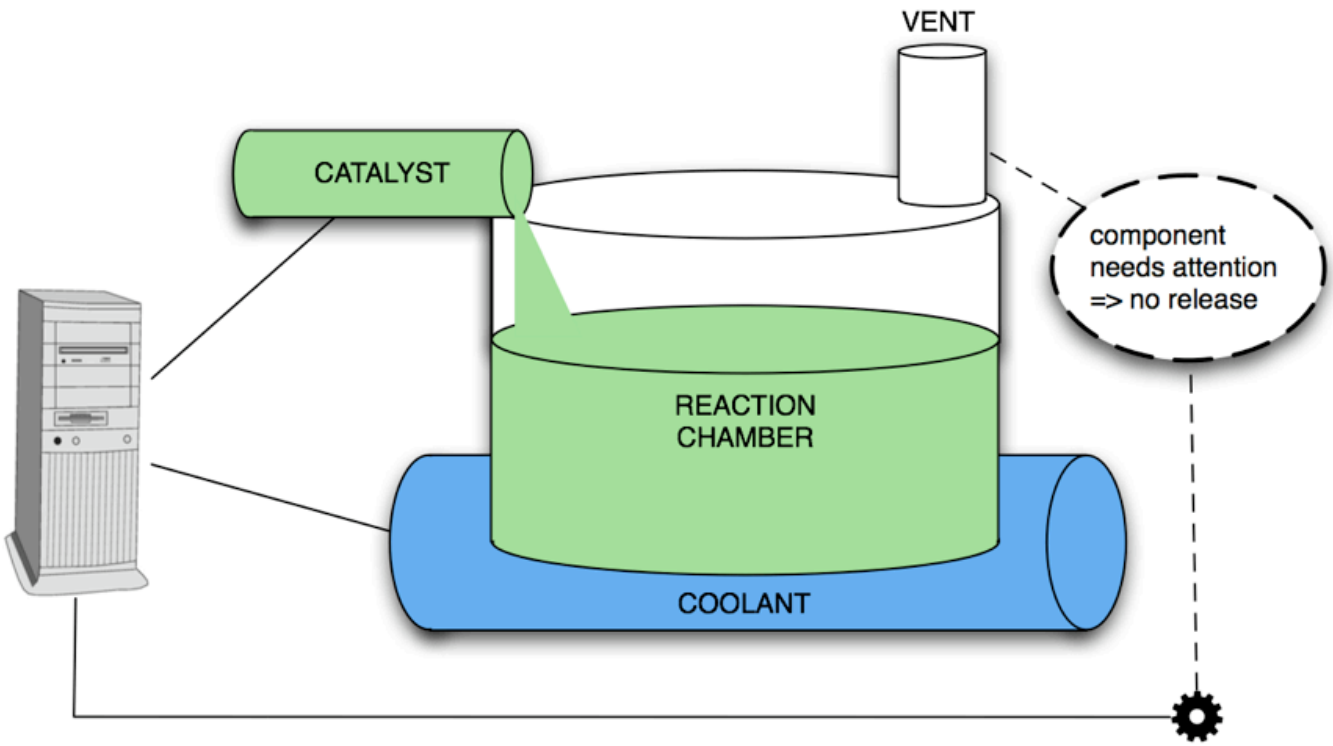
How it works


VENT
CATALYST
REACT
MODE
CAT open
level = 40%
REACTION
CHAMBER
COOL open
COOLANT

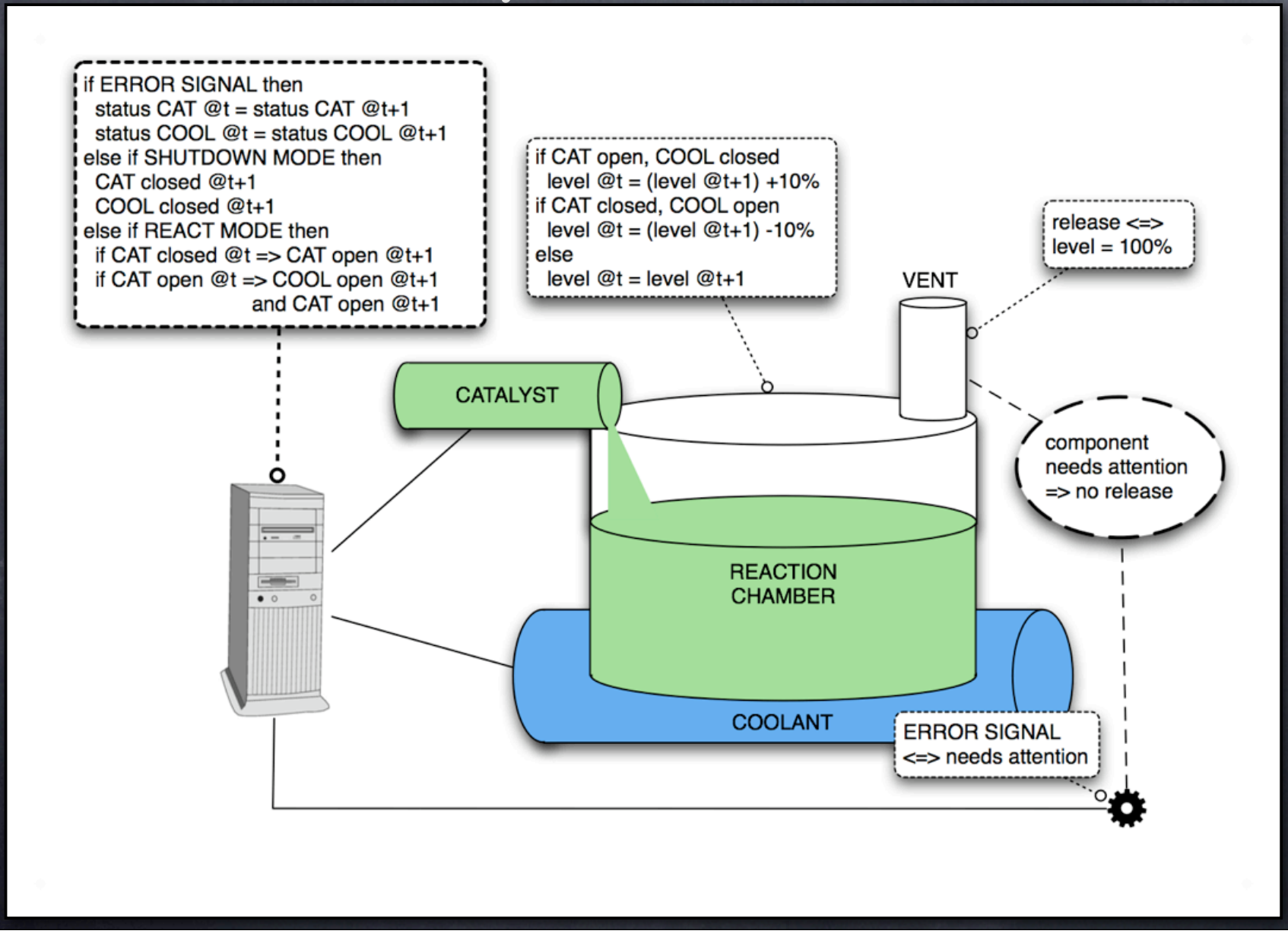
How it works


VENT
CATALYST
ERROR
MODE
CAT open
level = 40%
REACTION
CHAMBER
COOL open
COOLANT
ERROR SIGNAL
needs attention

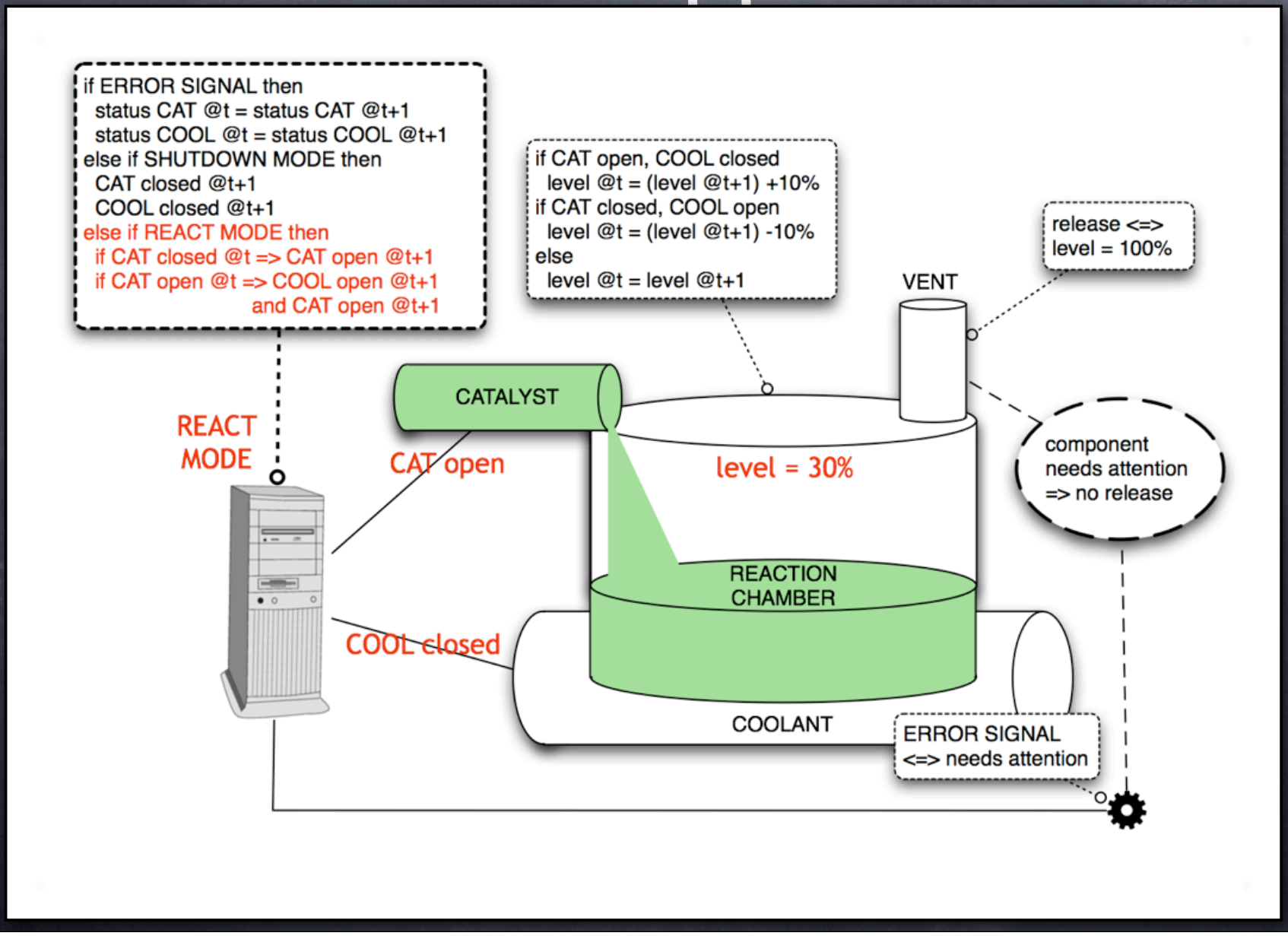
Why it works


VENT
CATALYST
component
needs attention
=> no release
REACTION
CHAMBER
COOLANT

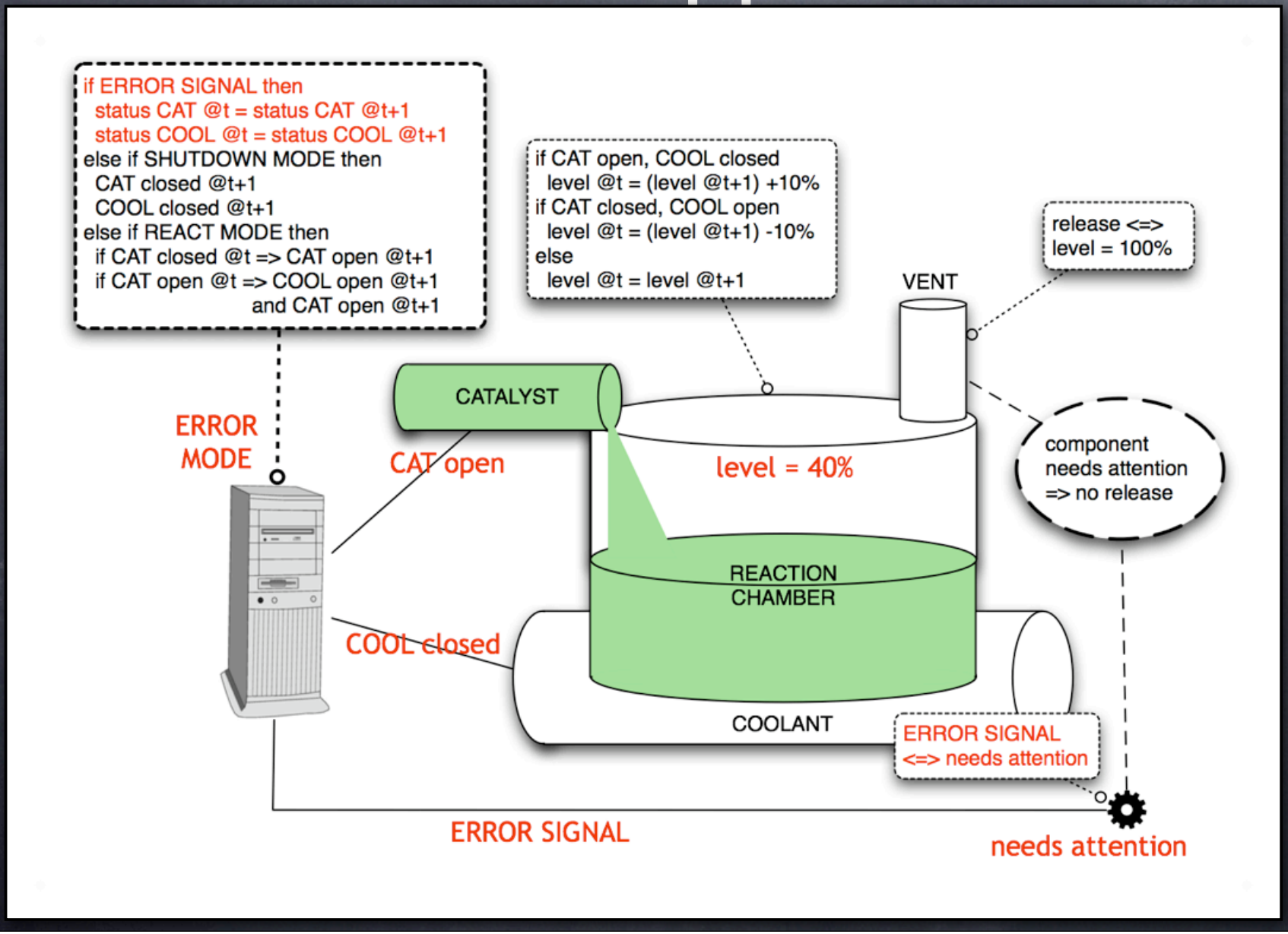
Why it works


if ERROR SIGNAL then
status CAT @t = status CAT @t+1
status COOL @t = status COOL @t+1
else if SHUTDOWN MODE then
CAT closed @t+1
COOL closed @t+1
else if REACT MODE then
if CAT closed @t => CAT open @t+1
if CAT open @t => COOL open @t+1
and CAT open @t+1
if CAT open, COOL closed
level @t = (level @t+1) +10%
if CAT closed, COOL open
level @t = (level @t+1) -10%
else
level @t = level @t+1
release <=>
level = 100%
VENT
CATALYST
component
needs attention
=> no release
REACTION
CHAMBER
COOLANT
ERROR SIGNAL
<=> needs attention

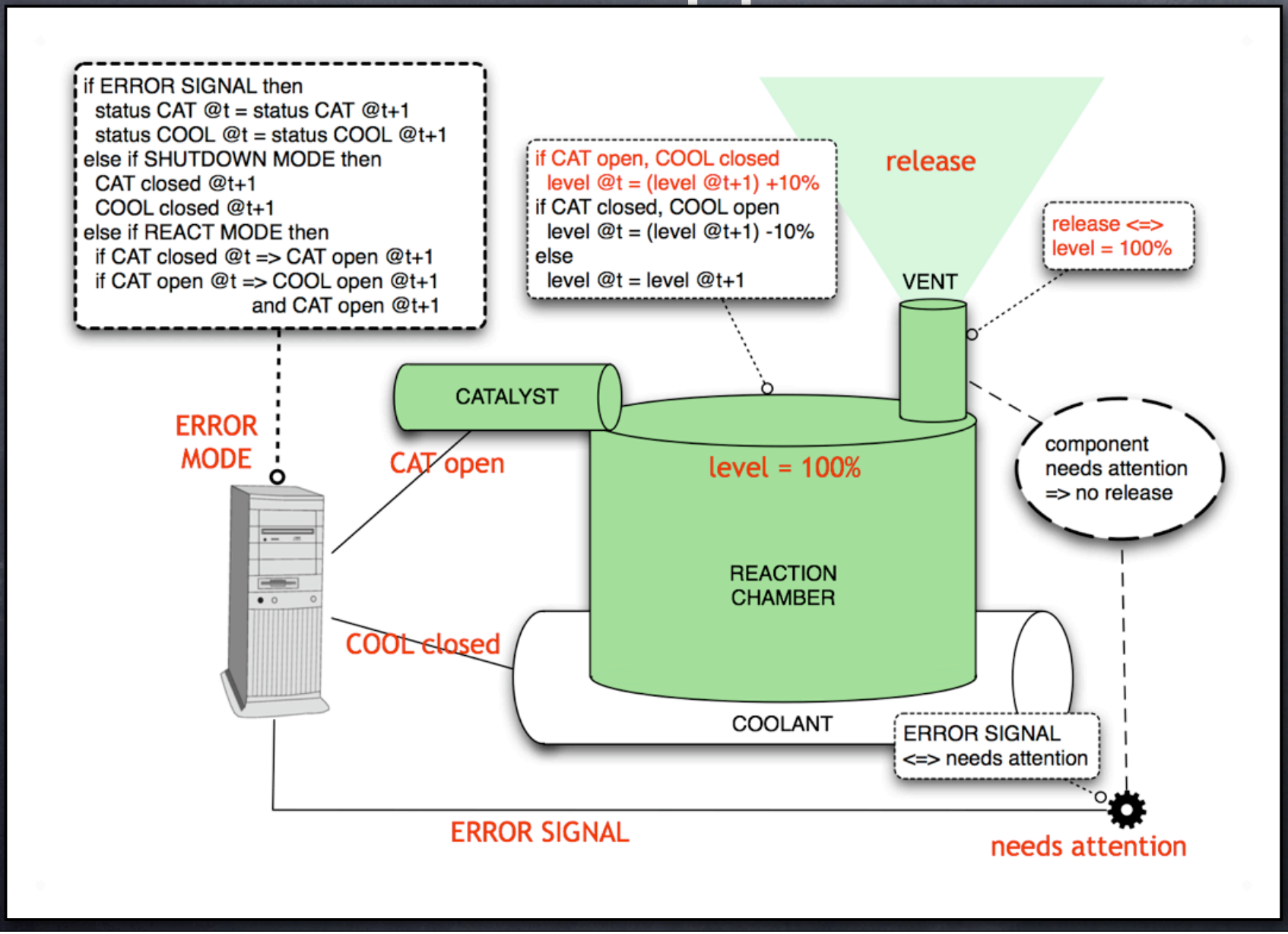
What happened



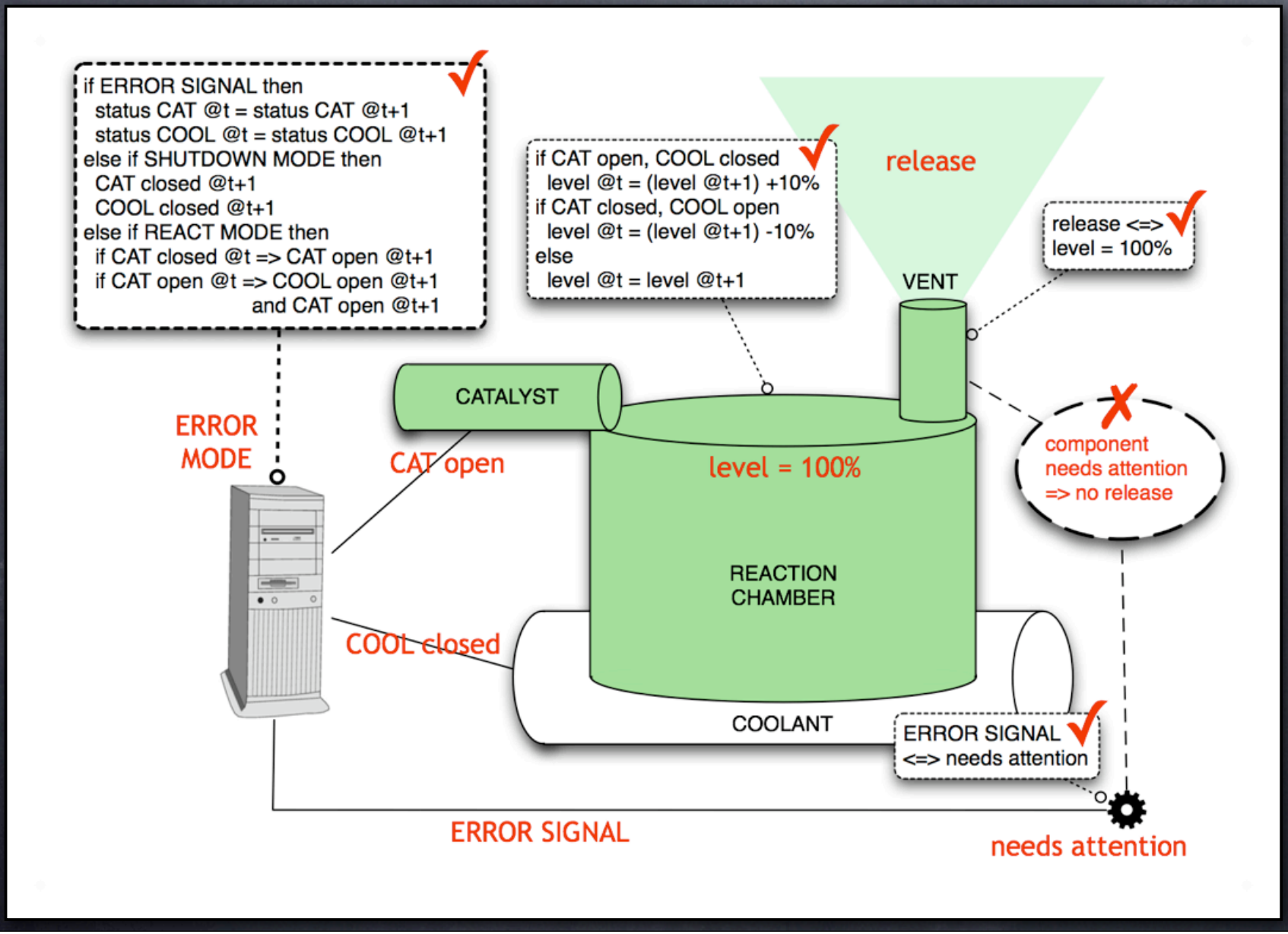
What happened



What happened



Who to blame?


if ERROR SIGNAL then
status CAT @t = status CAT @t+1
status COOL @t = status COOL @t+1
else if SHUTDOWN MODE then
CAT closed @t+1
COOL closed @t+1
else if REACT MODE then
if CAT closed @t => CAT open @t+1
if CAT open @t => COOL open @t+1
and CAT open @t+1
if CAT open, COOL closed
level @t = (level @t+1) +10%
if CAT closed, COOL open
level @t = (level @t+1) -10%
else
level @t = level @t+1
release
release <=>
level = 100%
VENT
CATALYST
ERROR
MODE
CAT open
level = 100%
component
needs attention
=> no release
REACTION
CHAMBER
COOL closed
COOLANT
ERROR SIGNAL
<=> needs attention
ERROR SIGNAL
needs attention

Who to blame?

what would have prevented the problem?

not a matter of better components

- redundant & resilient components
- software analysis & testing

not obvious which spec is wrong

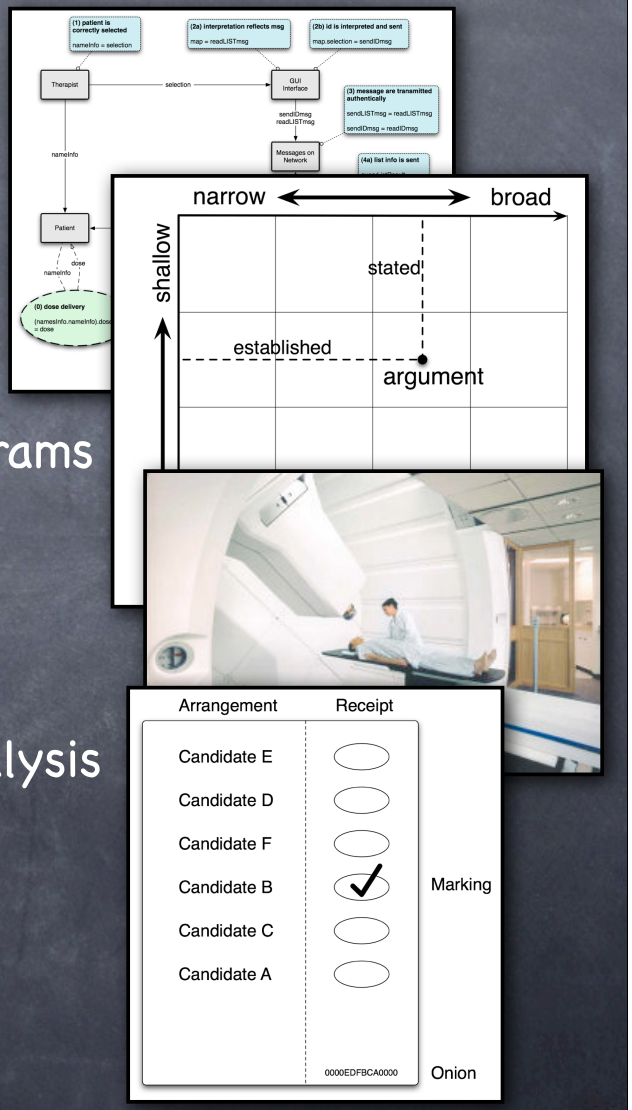
- SW should not simply halt on error
- tank should trigger coolant
- vent should pipe to enclosed tank

lack of system level understanding

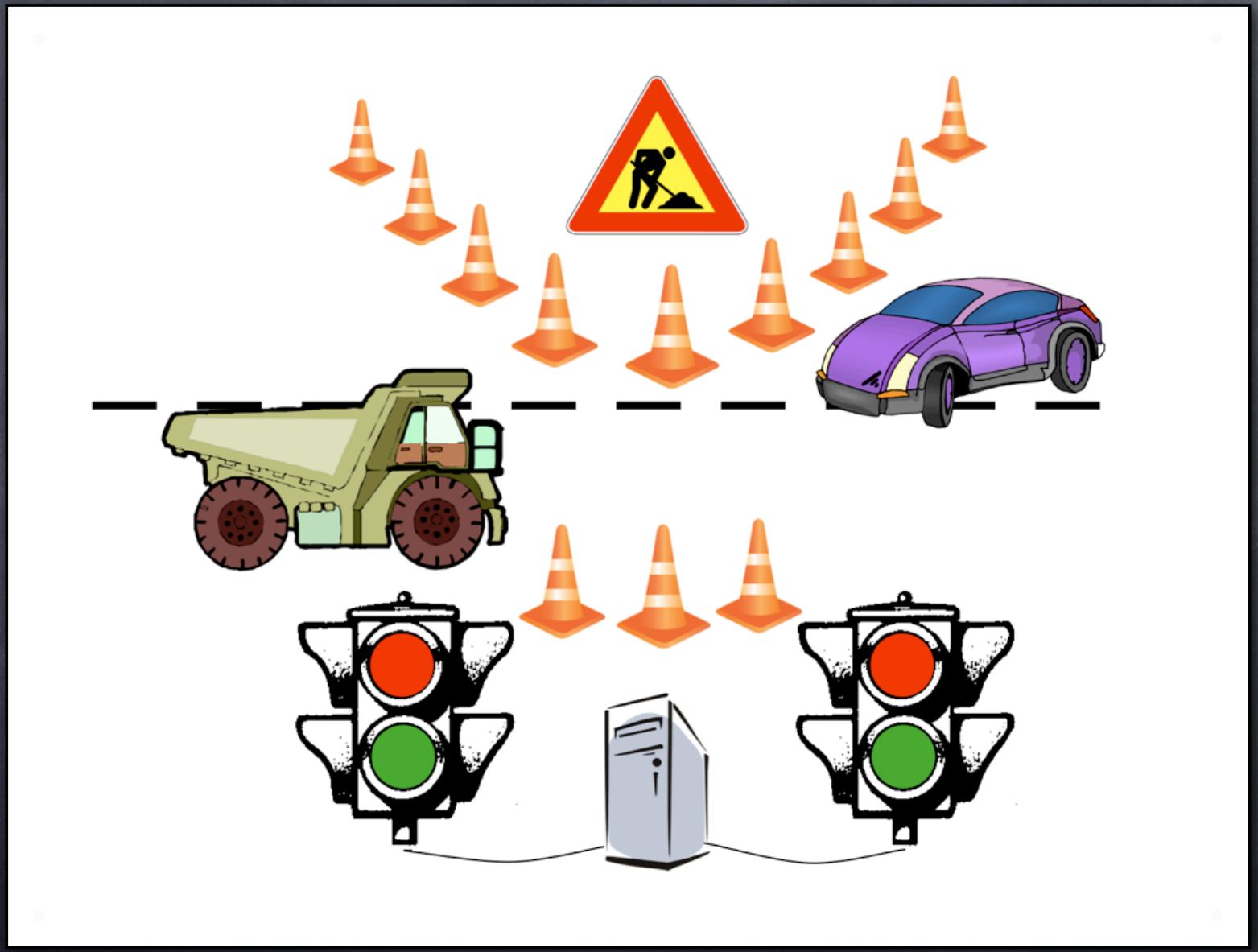
- undocumented assumptions
- poor system-level analysis

Contributions

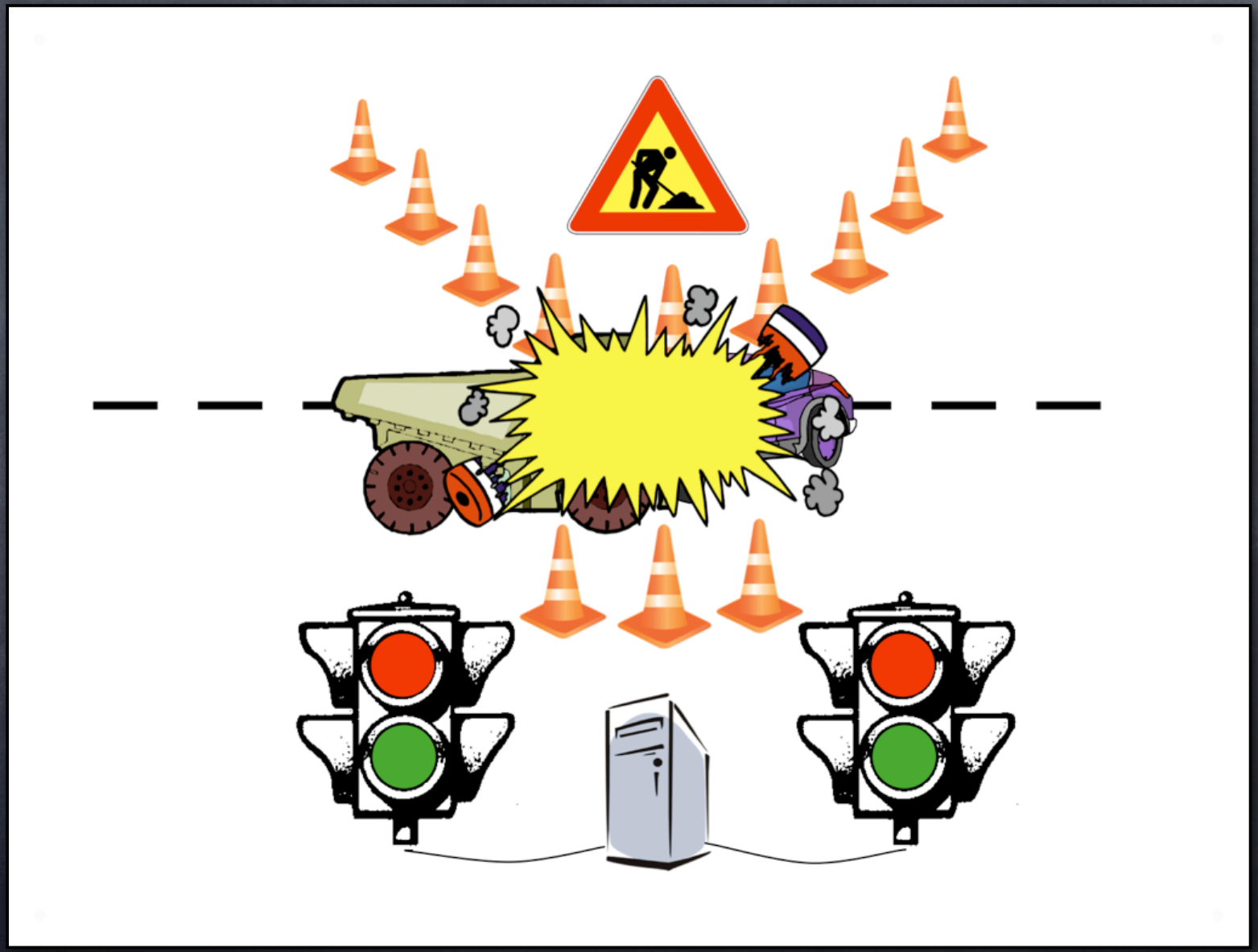
- Requirement Progression
  - systematically derive specifications from requirements
- CDAD framework
  - Composite Dependability Argument Diagrams structure end-to-end arguments
- 2 Case Studies
  - Proton Therapy
    - link requirement progression to code analysis
  - Electronic Voting
    - link fidelity to secrecy, auditability



Traffic Light



Traffic Light



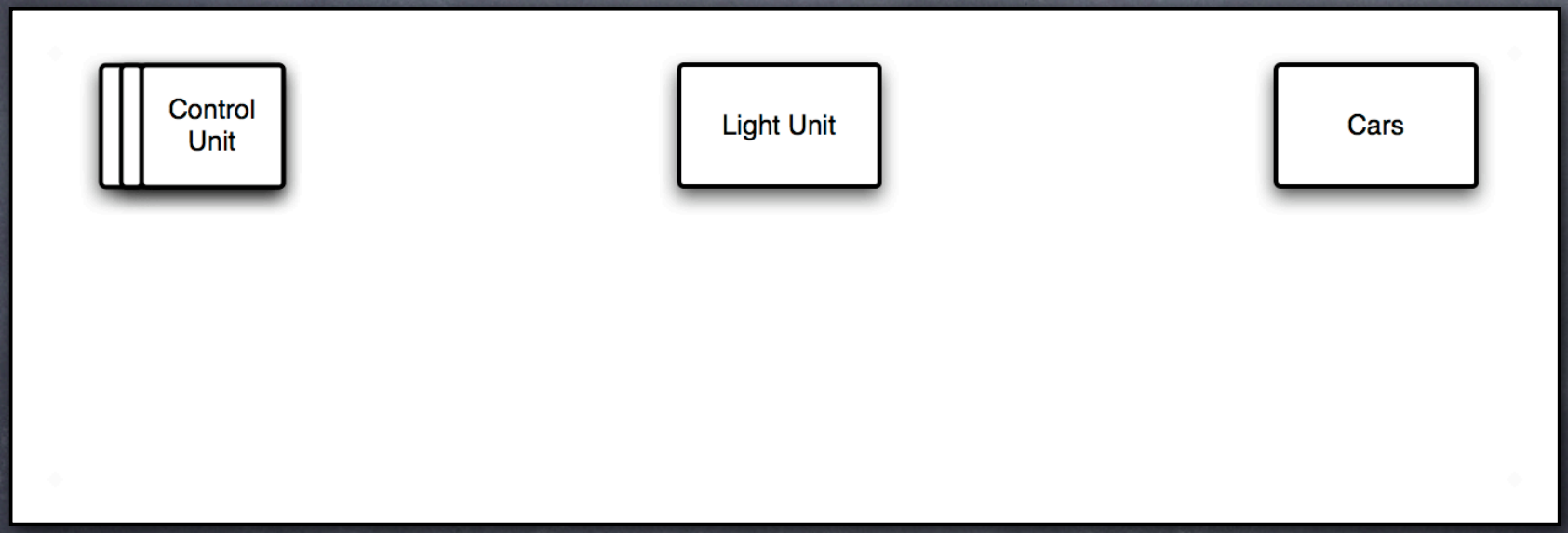
Traffic Light

Control Unit

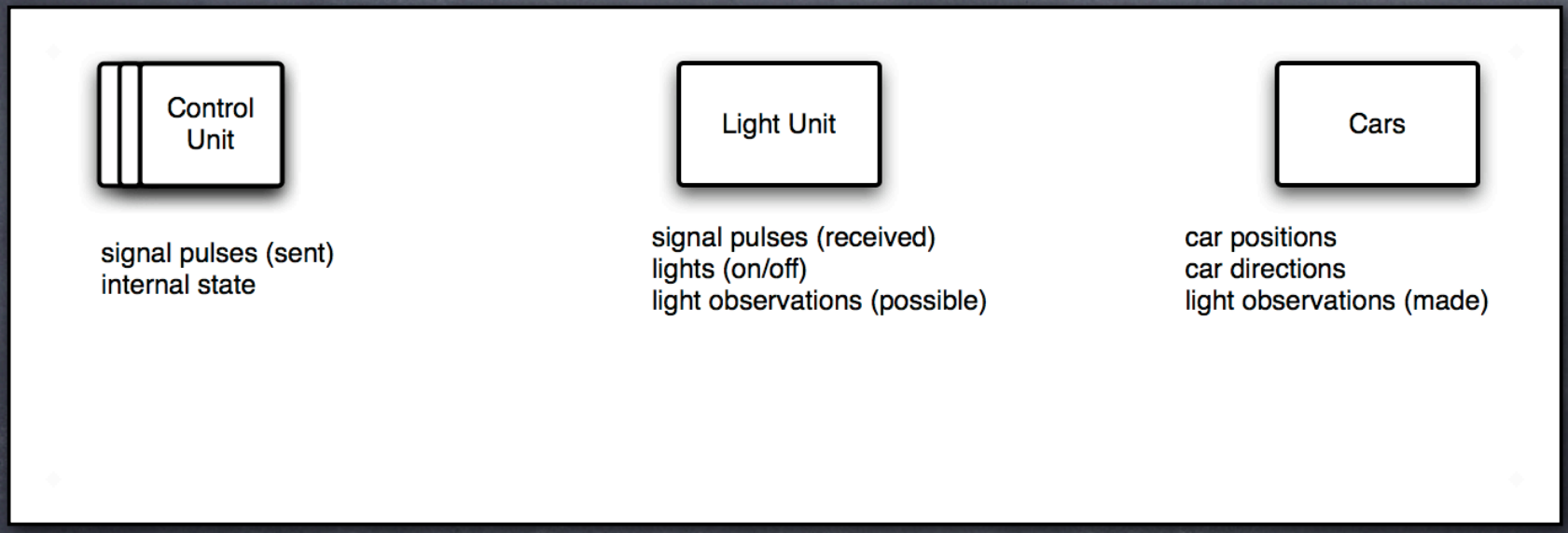
Light Unit

Cars

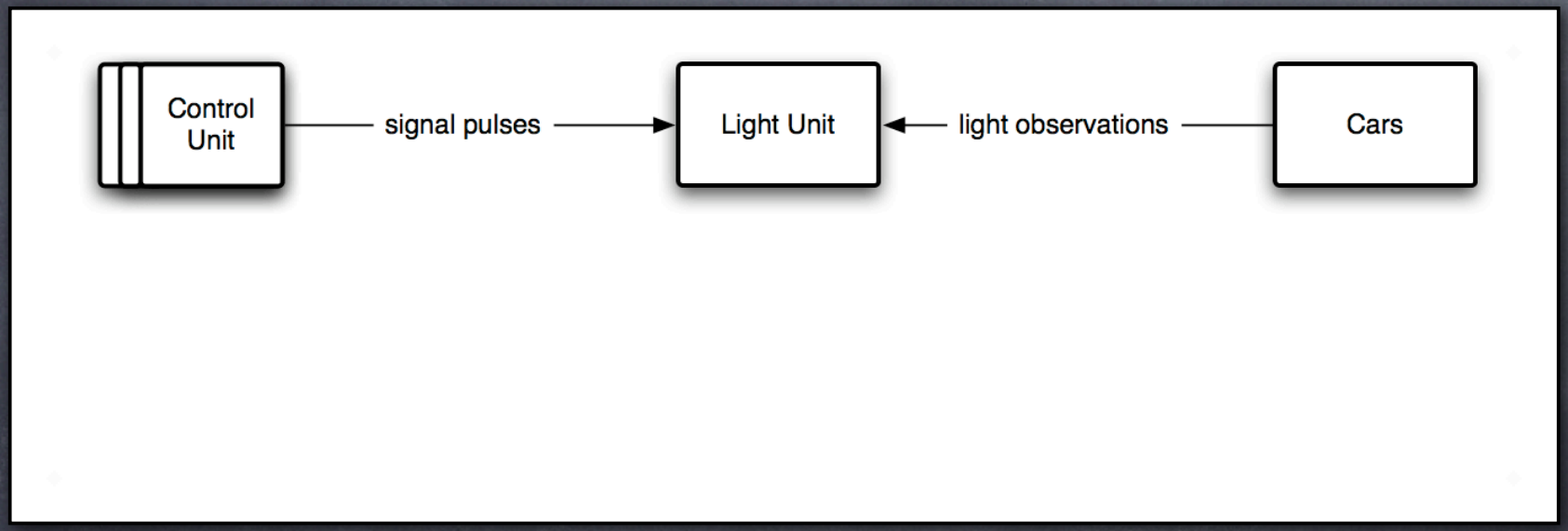
Traffic Light Domains



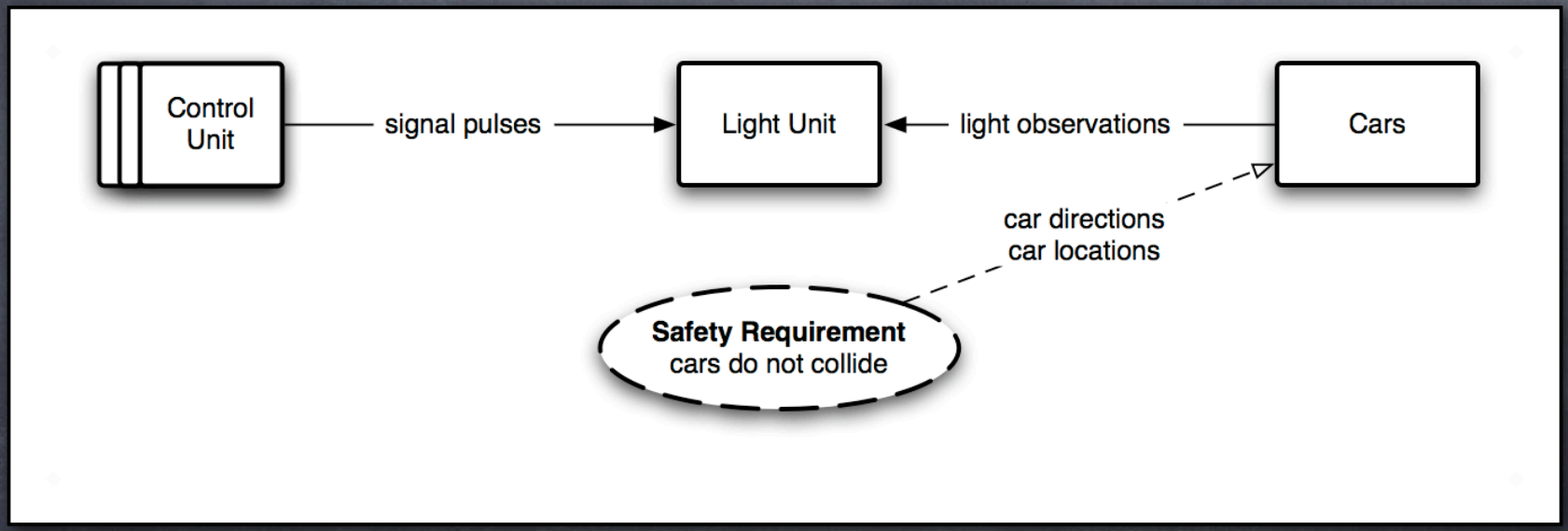
Traffic Light Phenomena



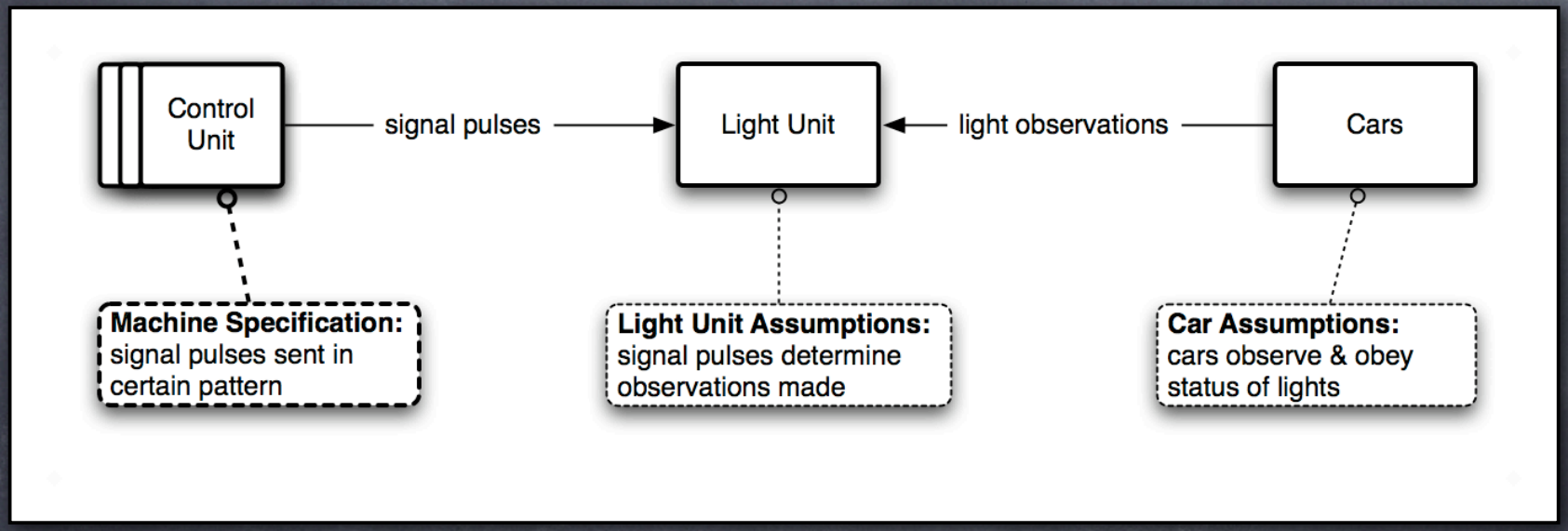
Traffic Light Shared Phenomena



Traffic Light Requirement



Traffic Light Spec & Assumptions



What if something is missing?

identify elements of implication

- Machine Spec ∧ Light Unit Assumptions ∧ Car Assumptions ⇒ Requirement

what if something is missing?

- nothing missing: check implication
- one missing: identify weakest
- several missing: decompose

What is typically done?

unstructured approach

- put engineers in room together
- document the resulting design

resulting specification document is

- too vague - missing explanations
- too weak - implicit assumptions
- too strong - implementation bias

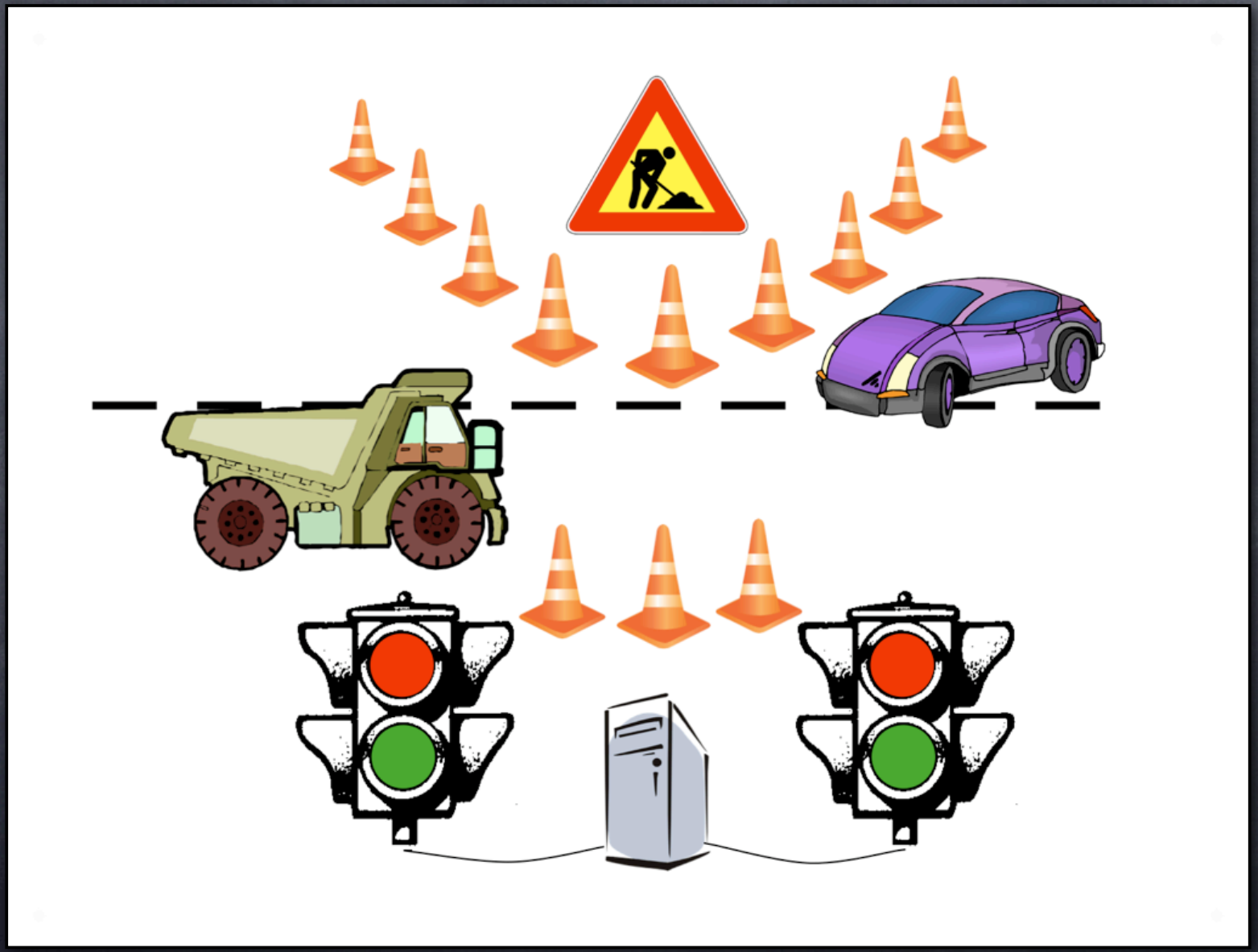
A new approach: requirement progression

systematic process to guide **human** proposing meaningful assumptions

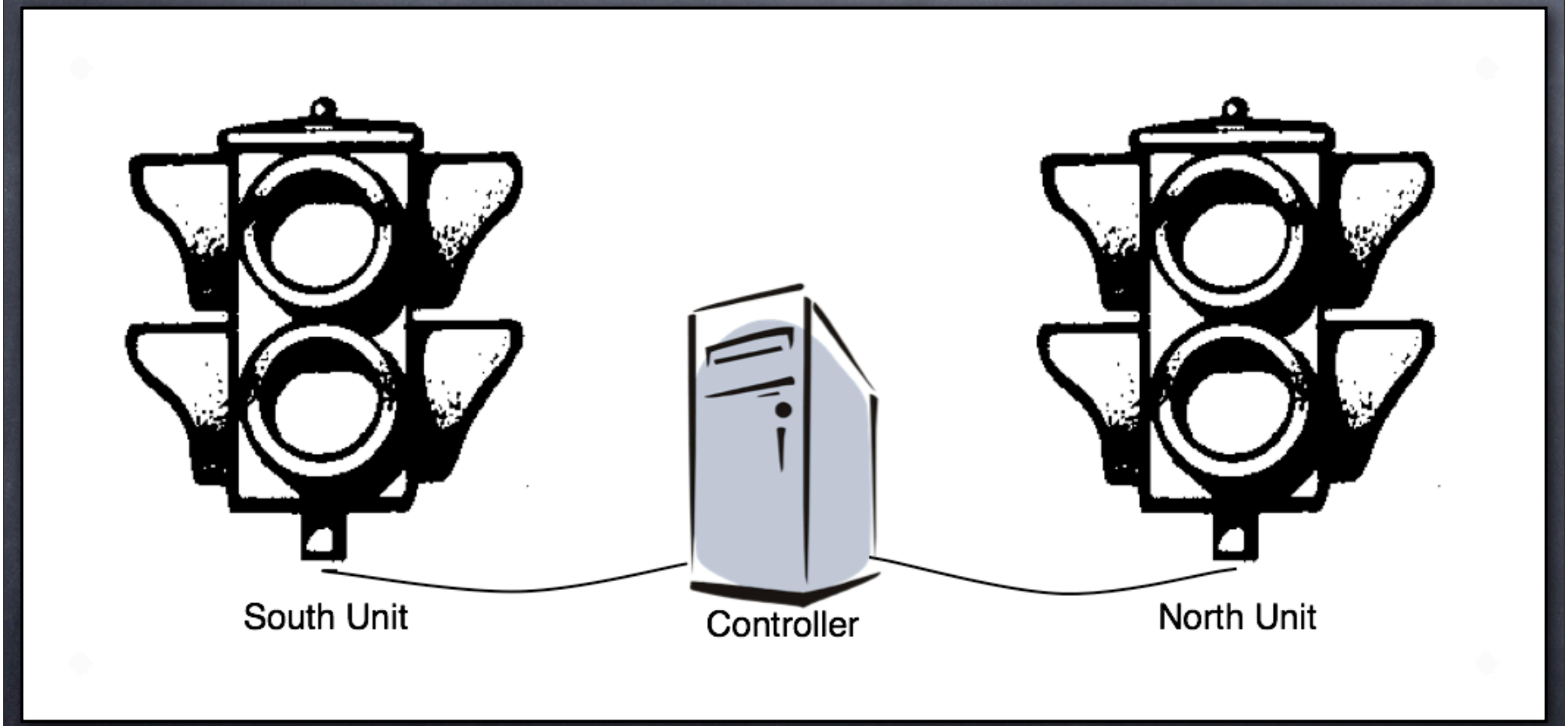
explicitly document assumptions using **precise** language

automatically check proposals

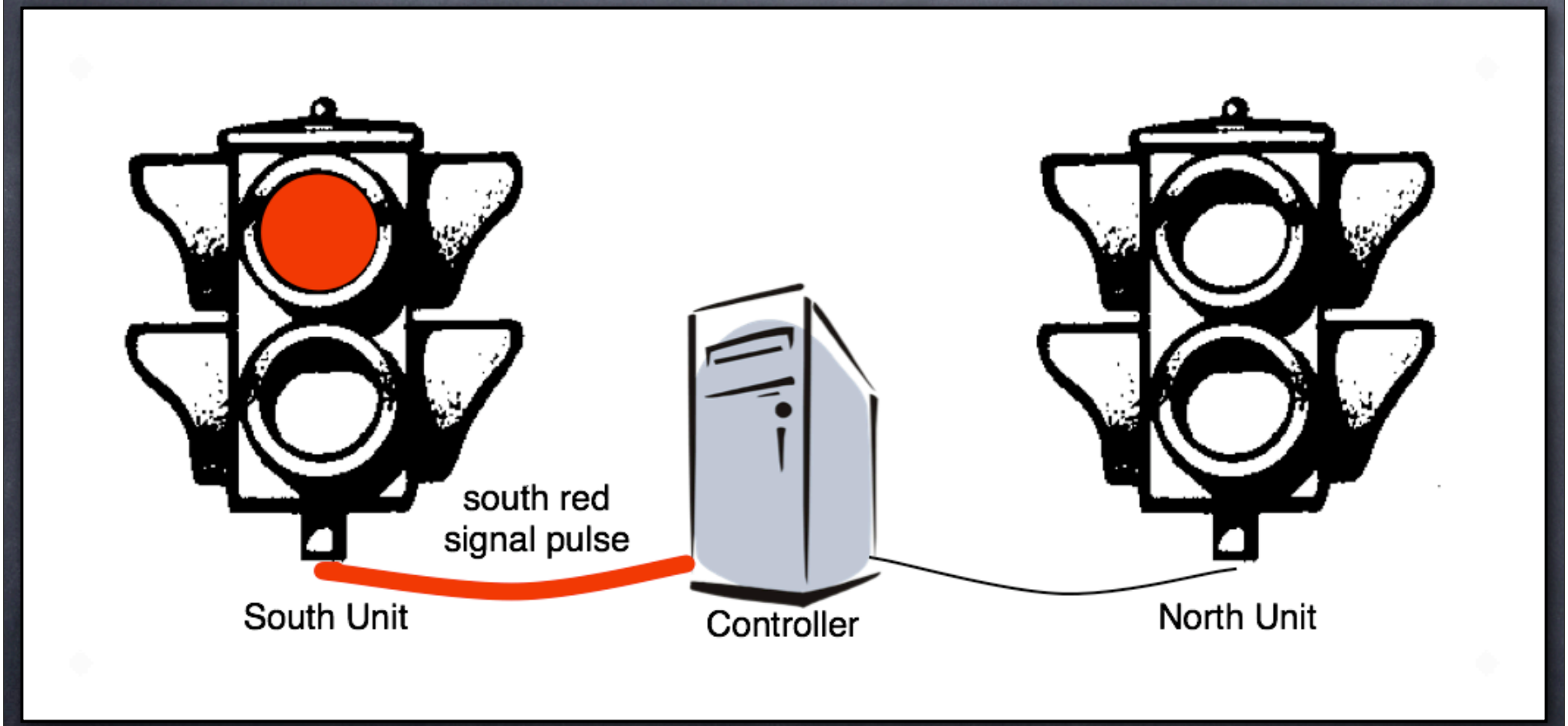
Traffic Light Revisited



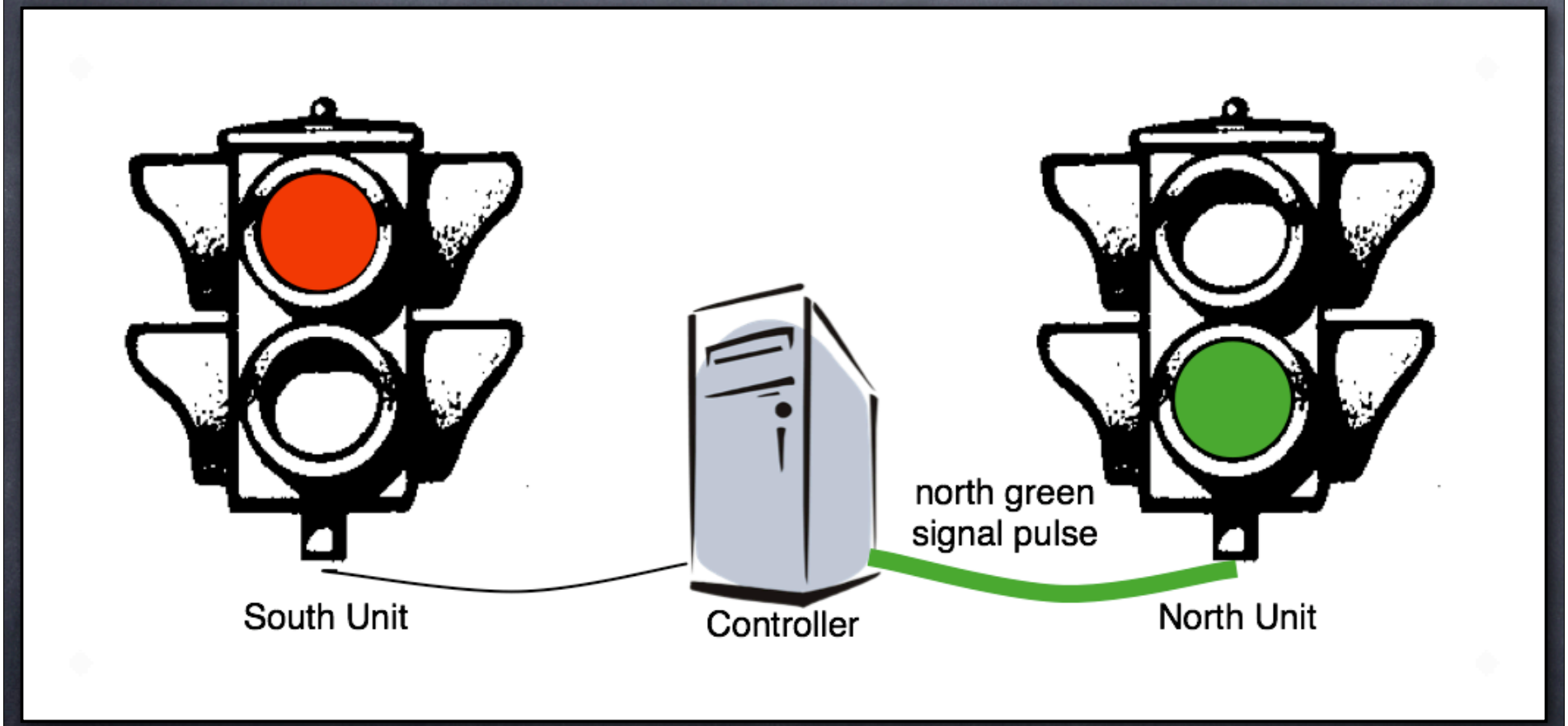
Light Unit Behavior



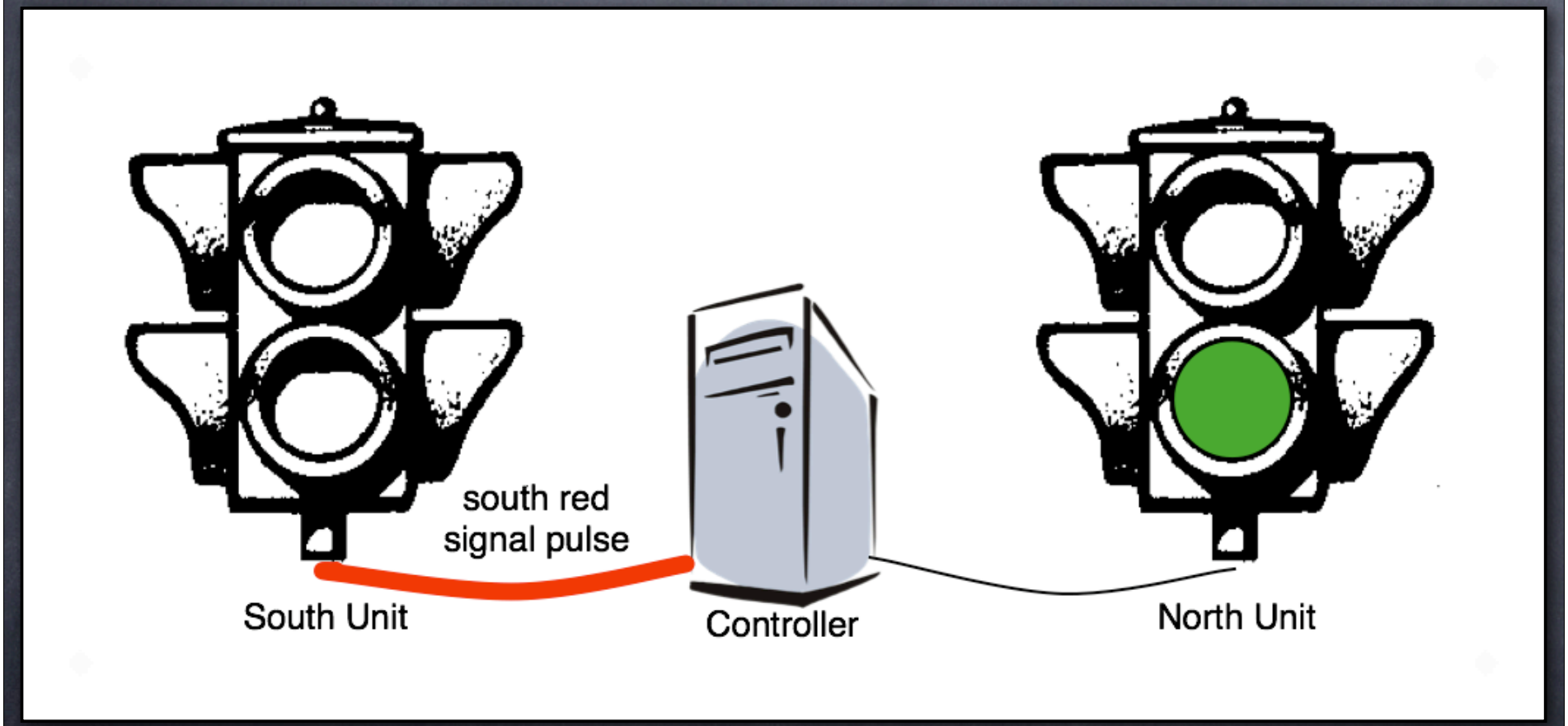
Light Unit Behavior



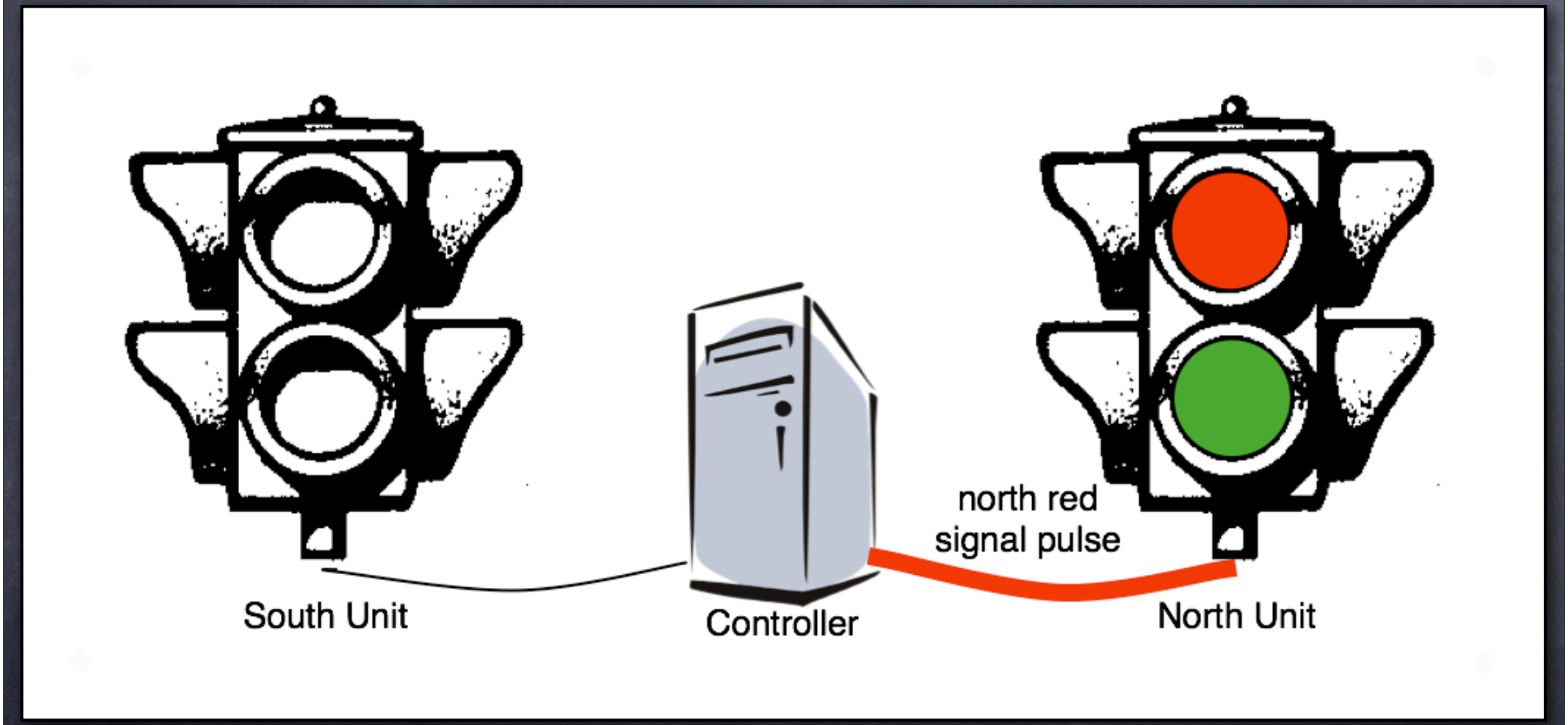
Light Unit Behavior



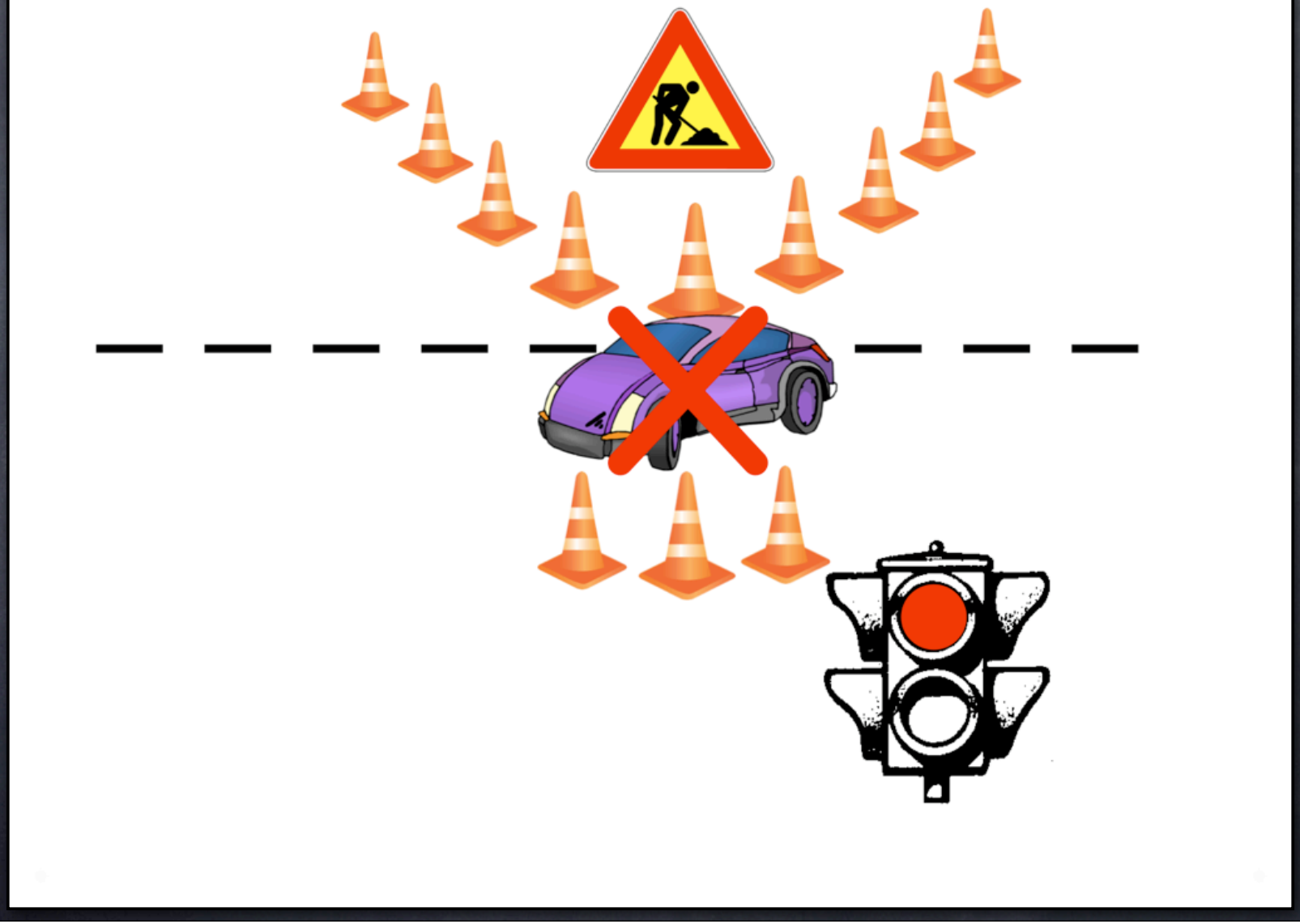
Light Unit Behavior



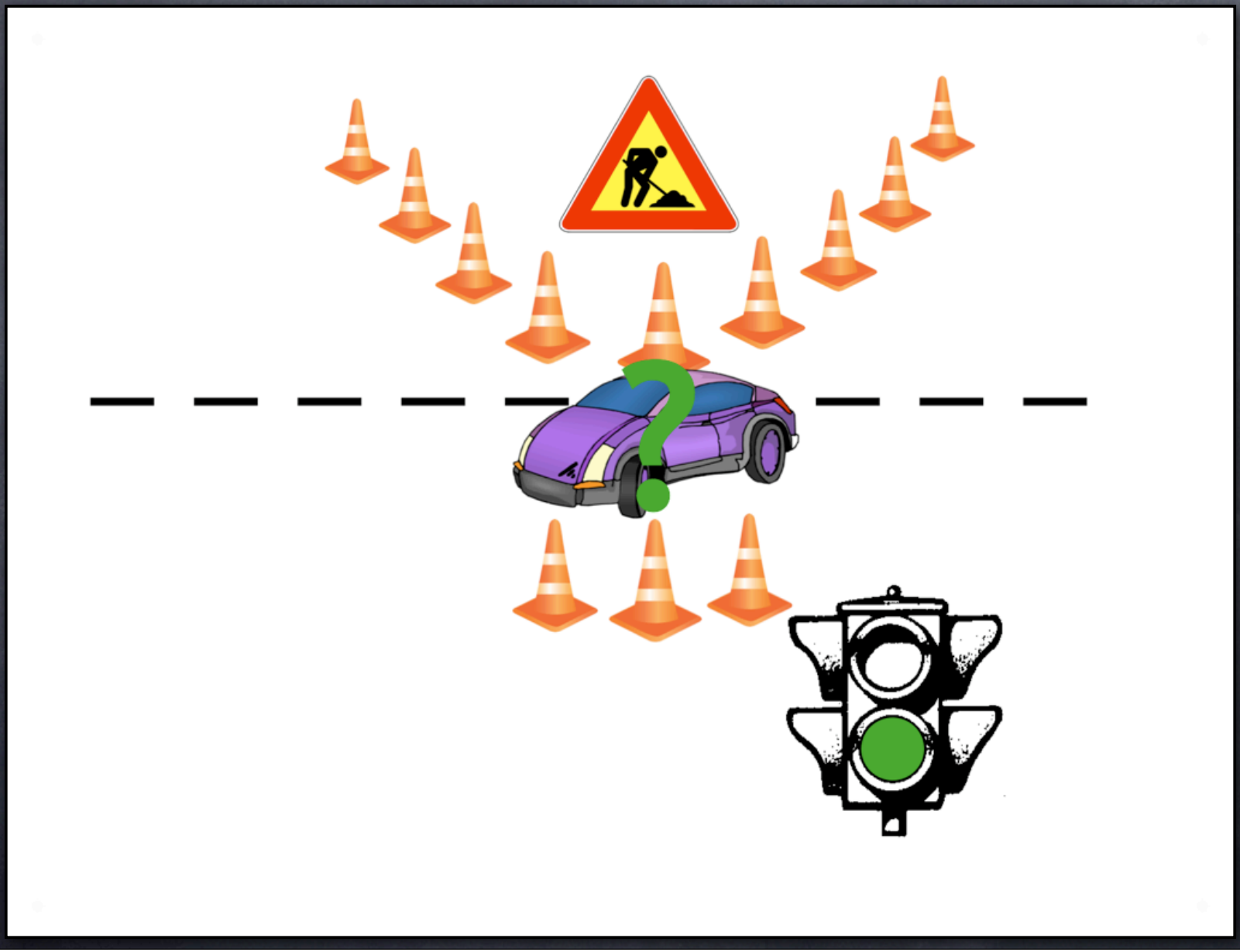
Light Unit Behavior



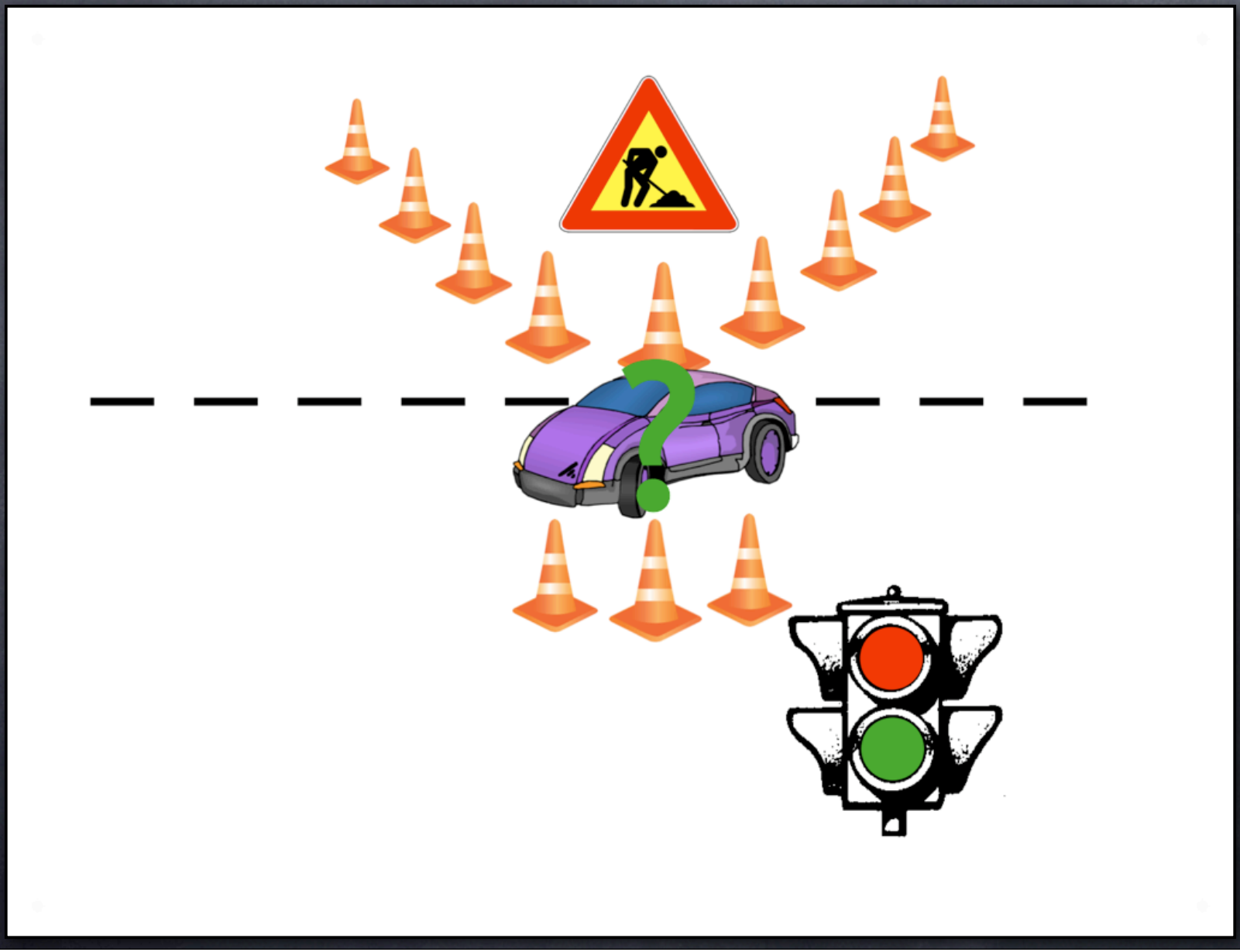
Car Behaviors



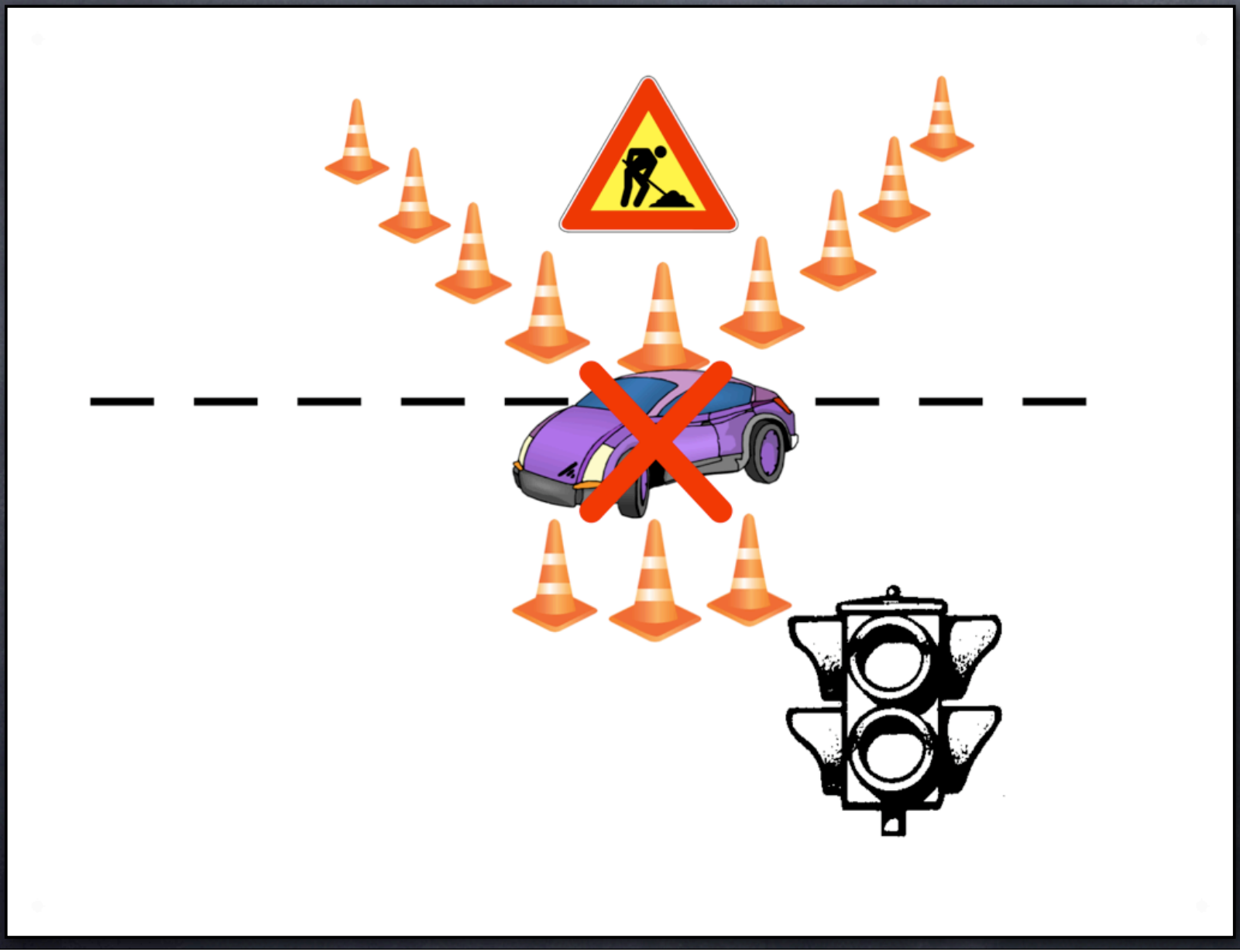
Car Behaviors



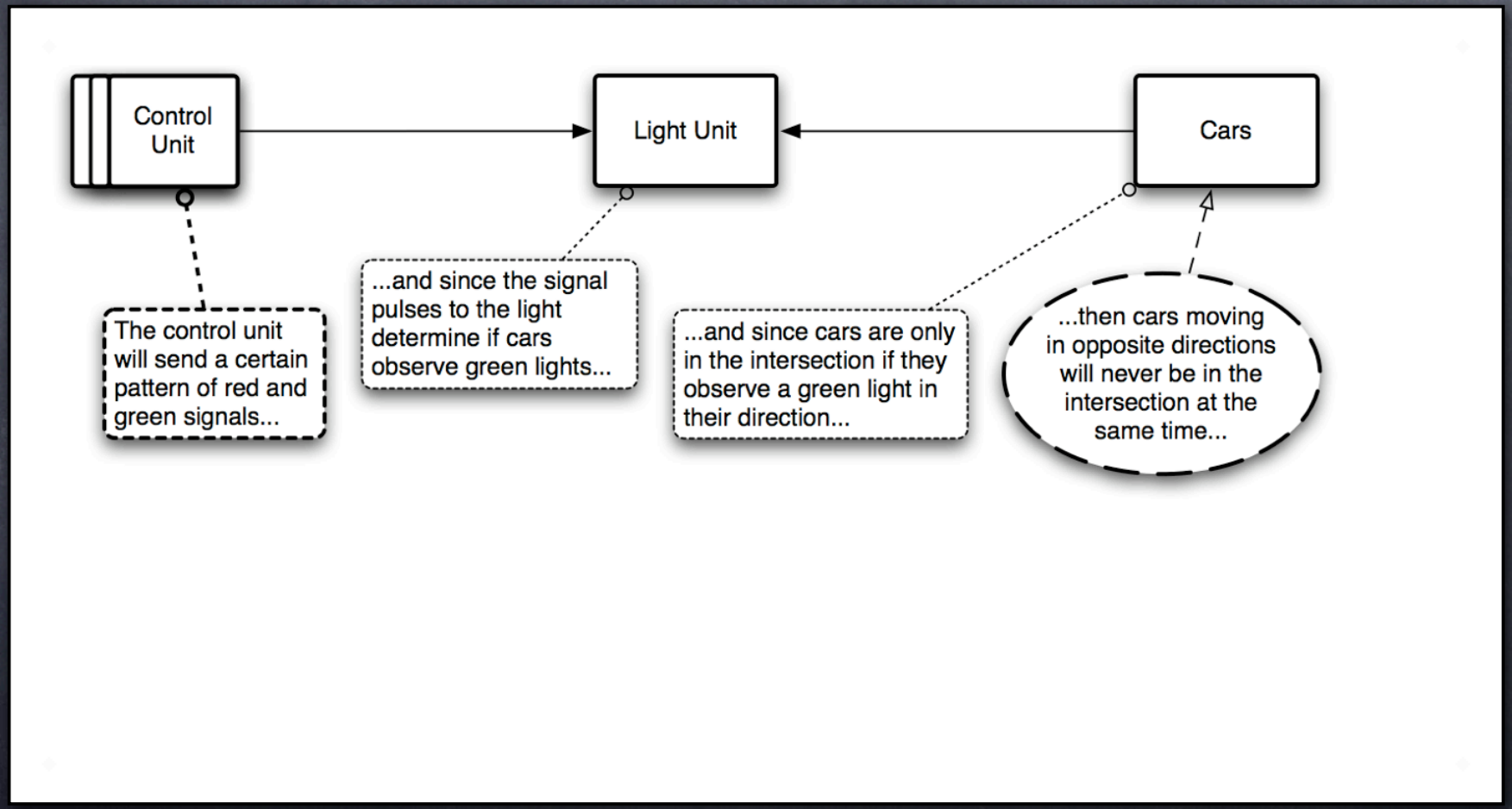
Car Behaviors



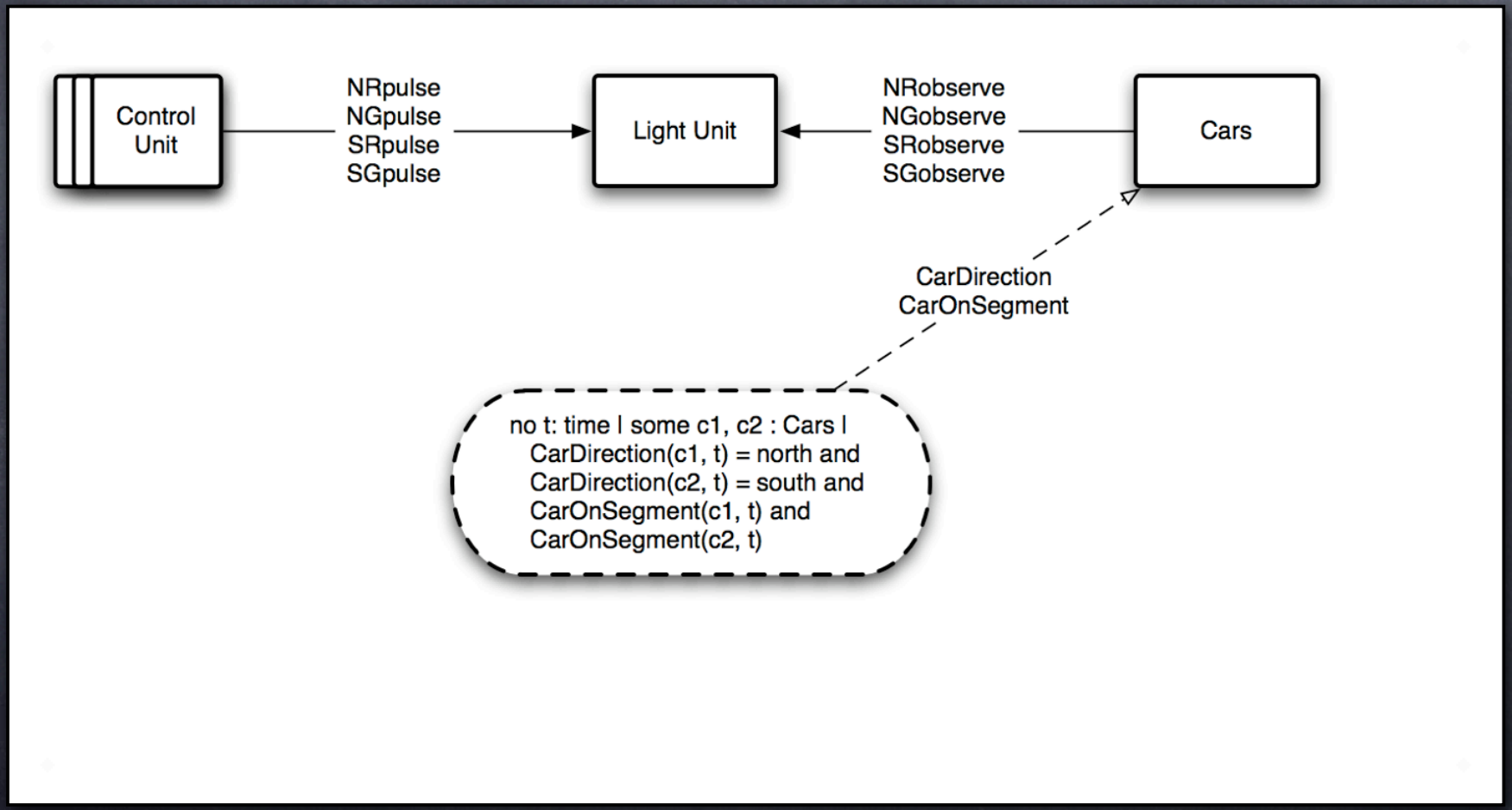
Car Behaviors



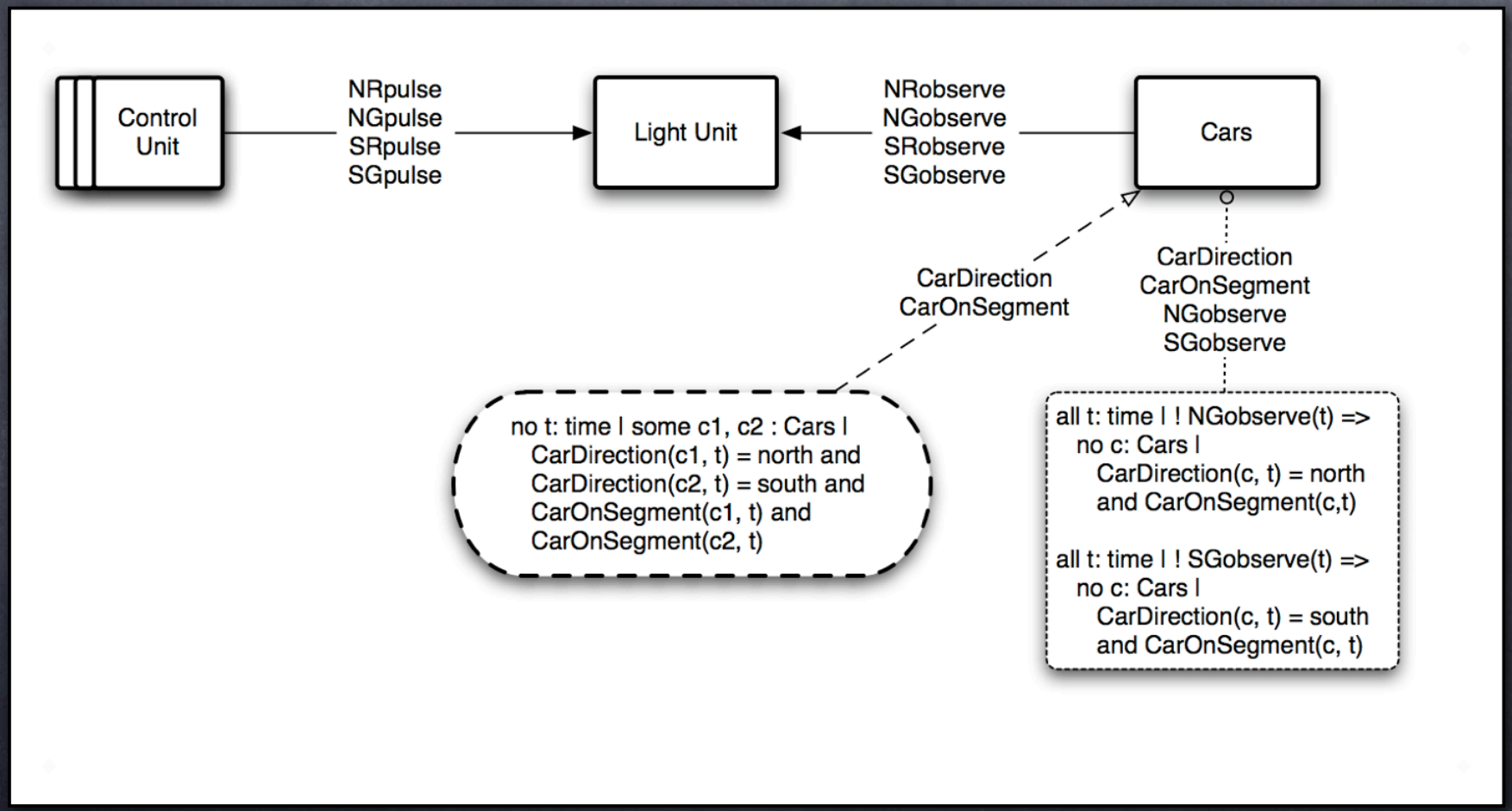
Requirement Progression frame diagram



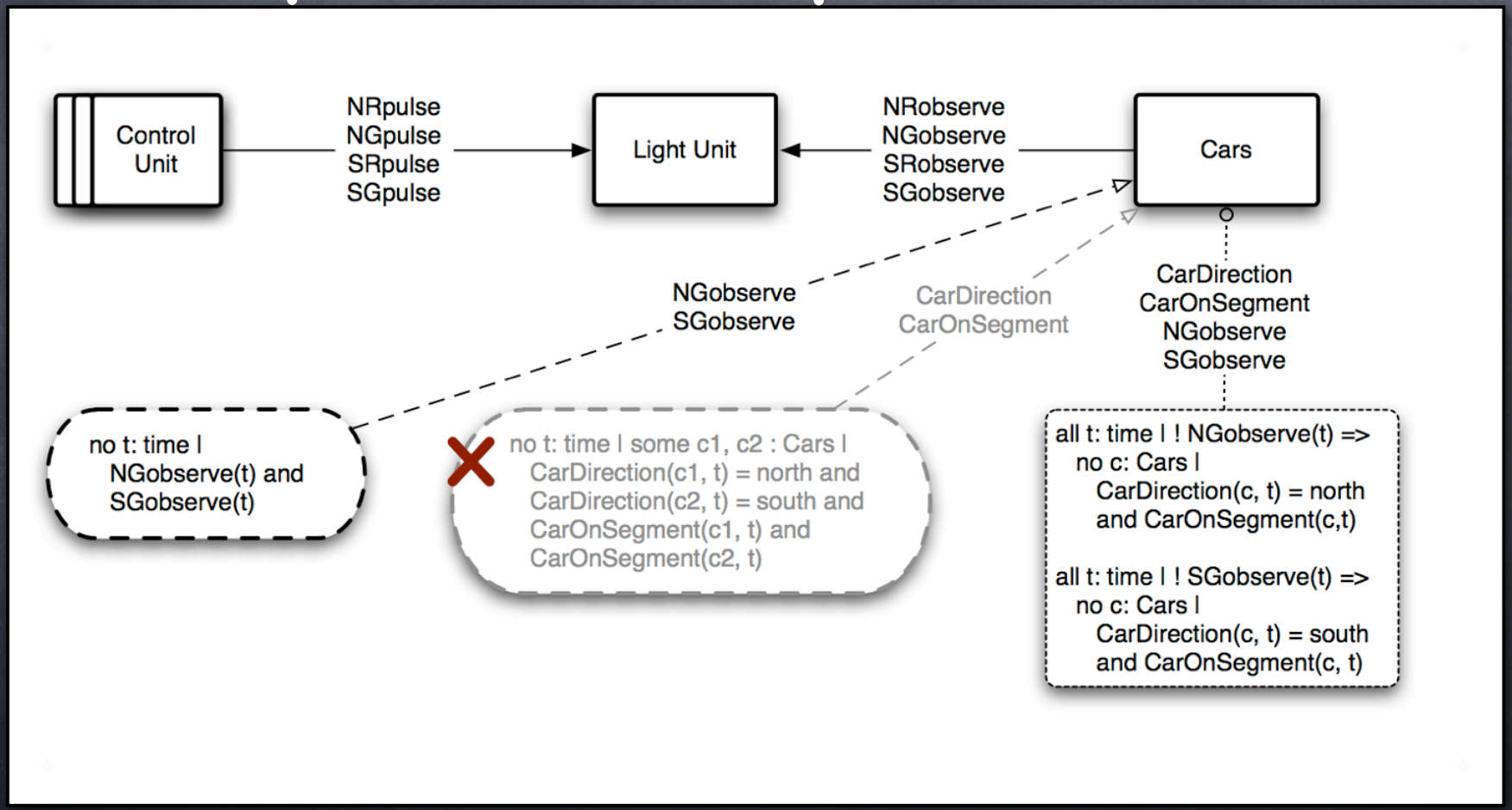
Requirement Progression formal requirement



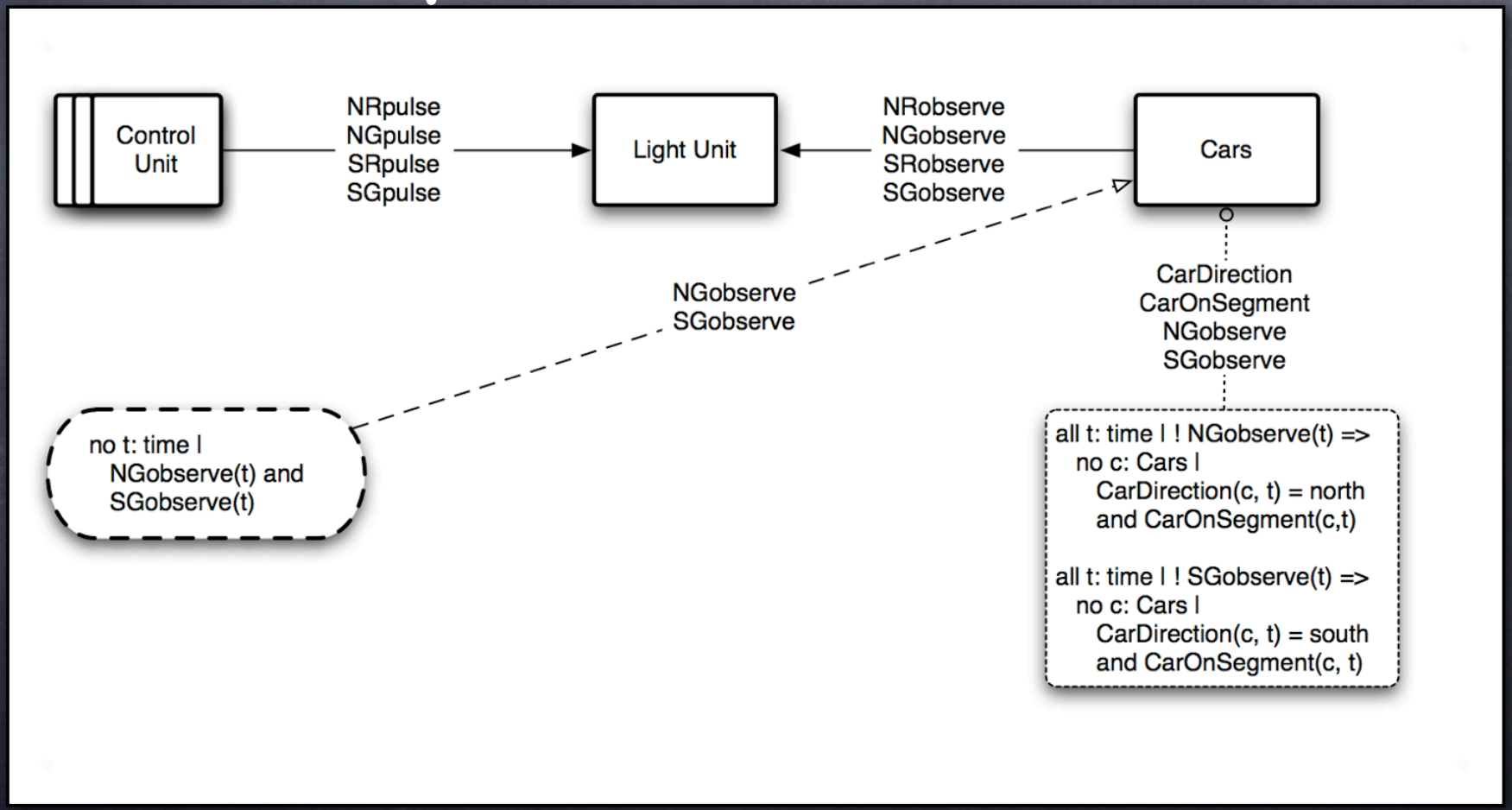
Requirement Progression add domain assumption



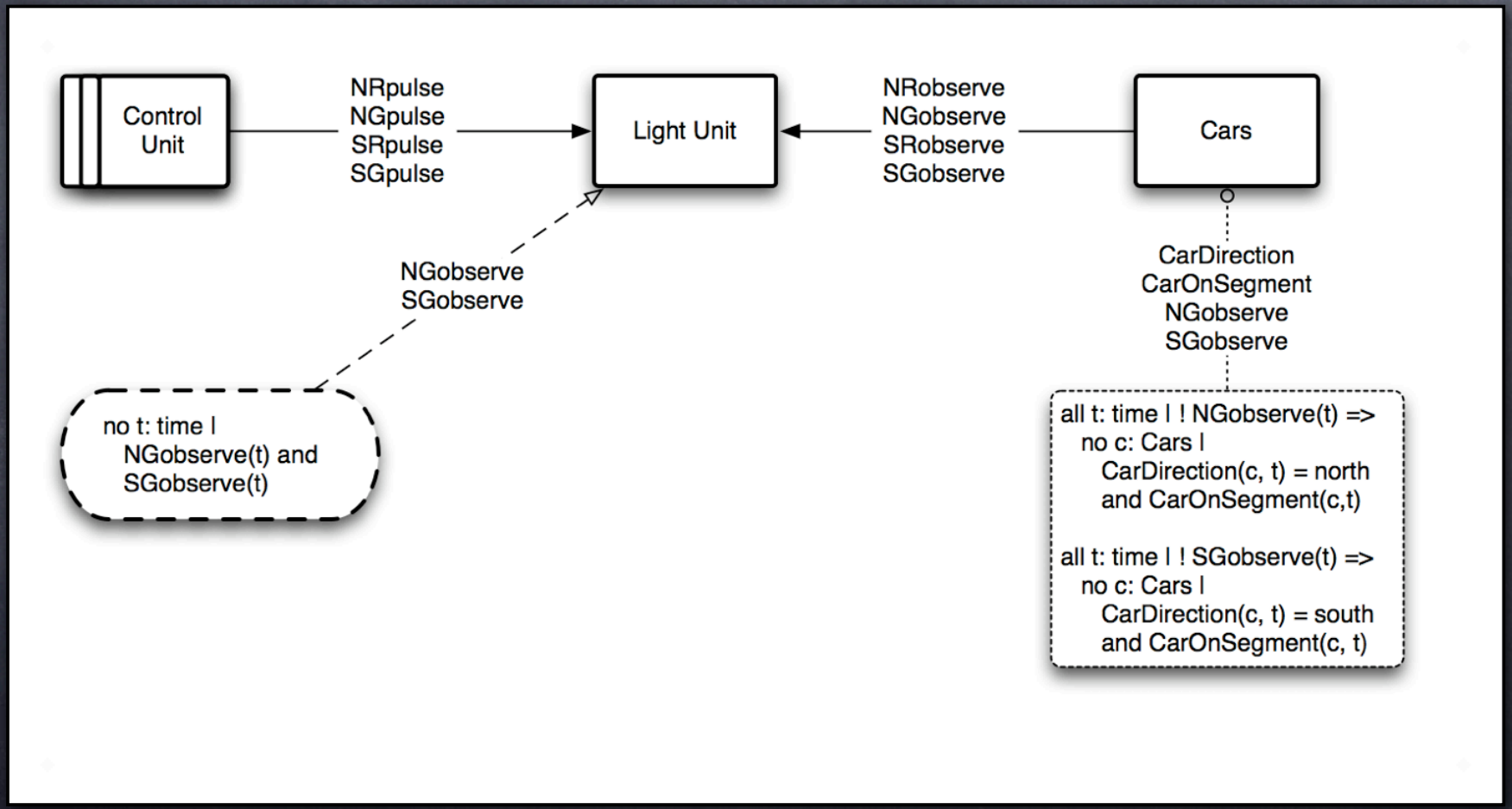
Requirement Progression rephrase requirement



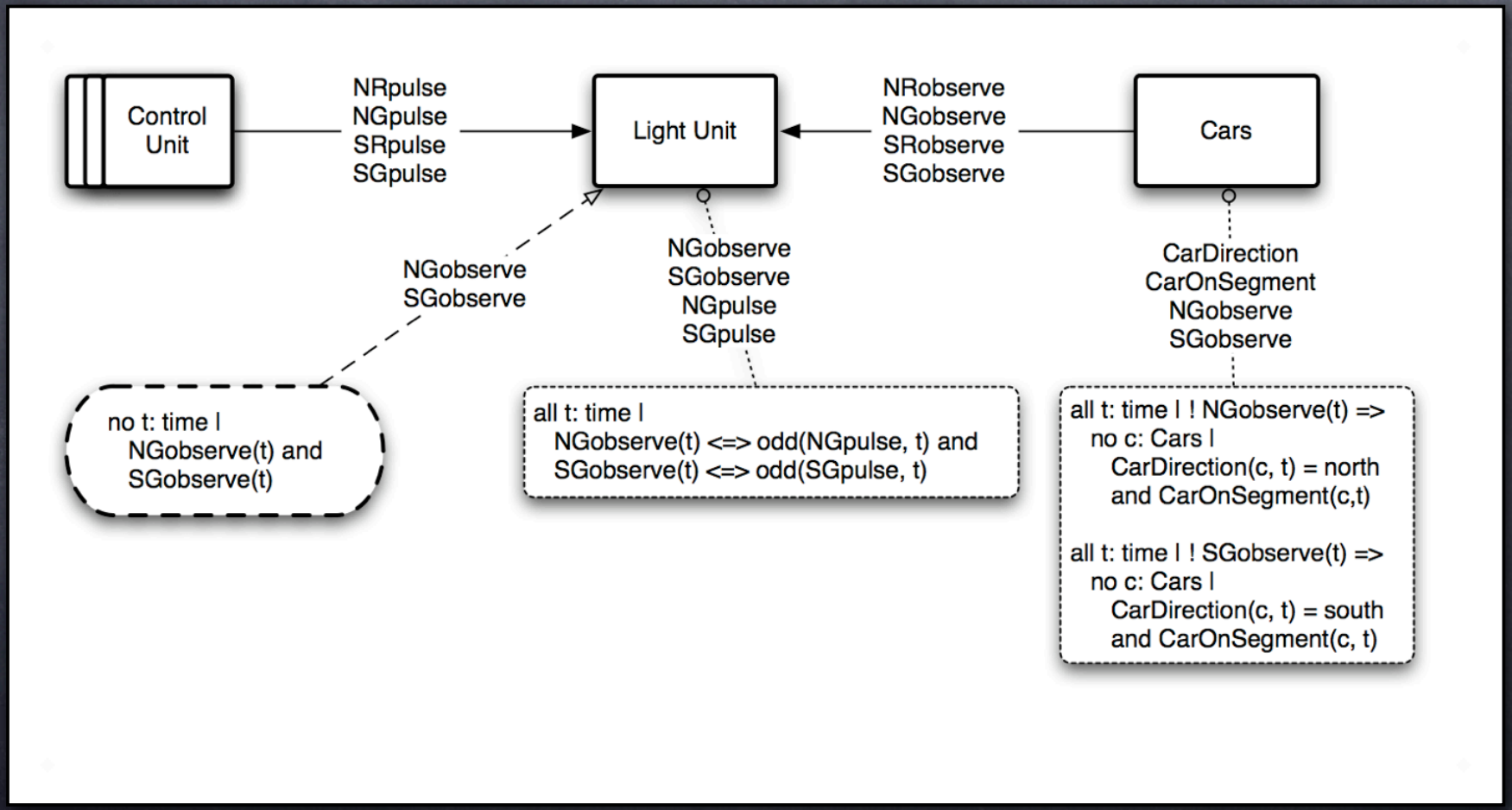
Requirement Progression implication holds



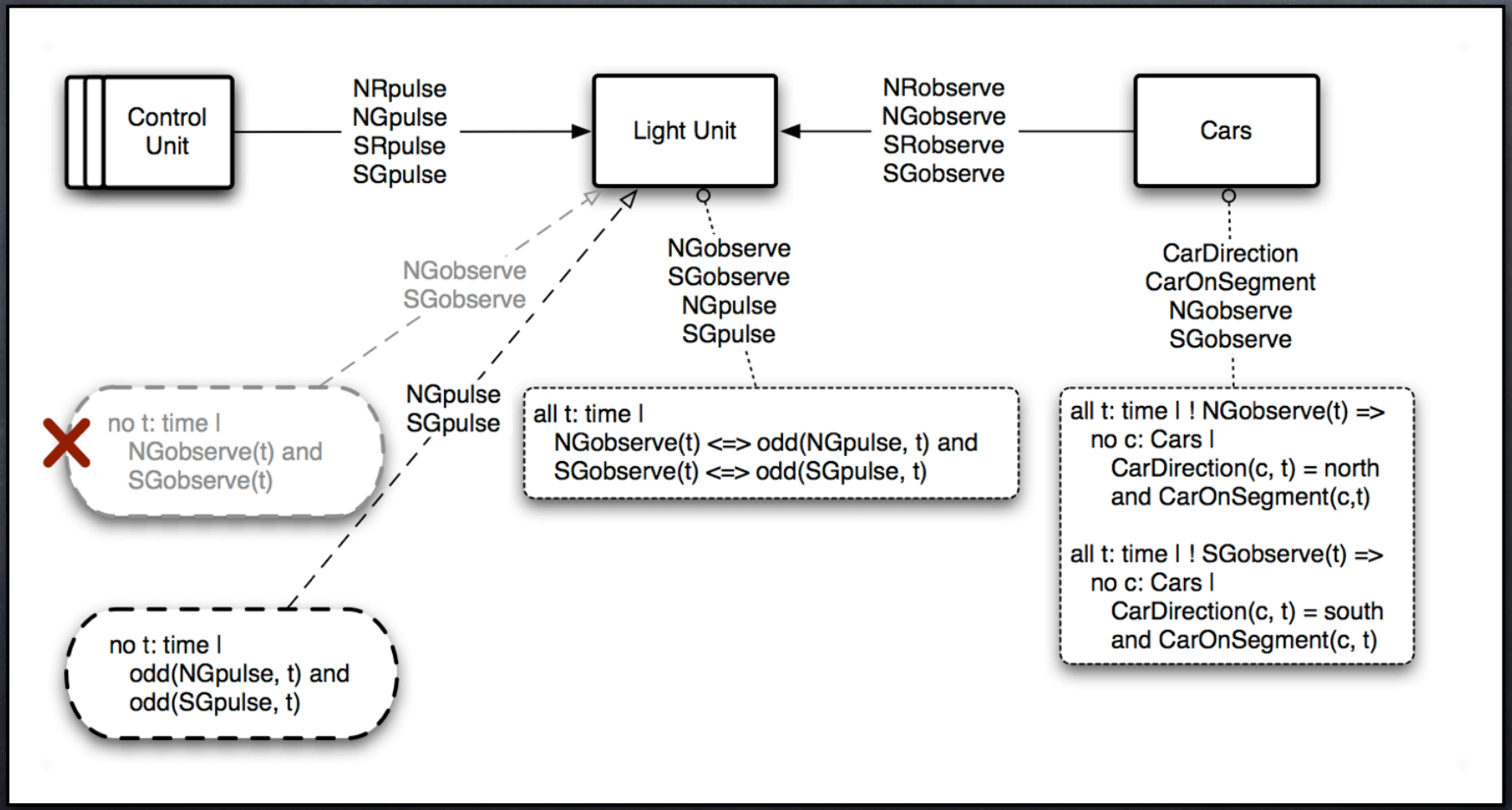
Requirement Progression
push requirement



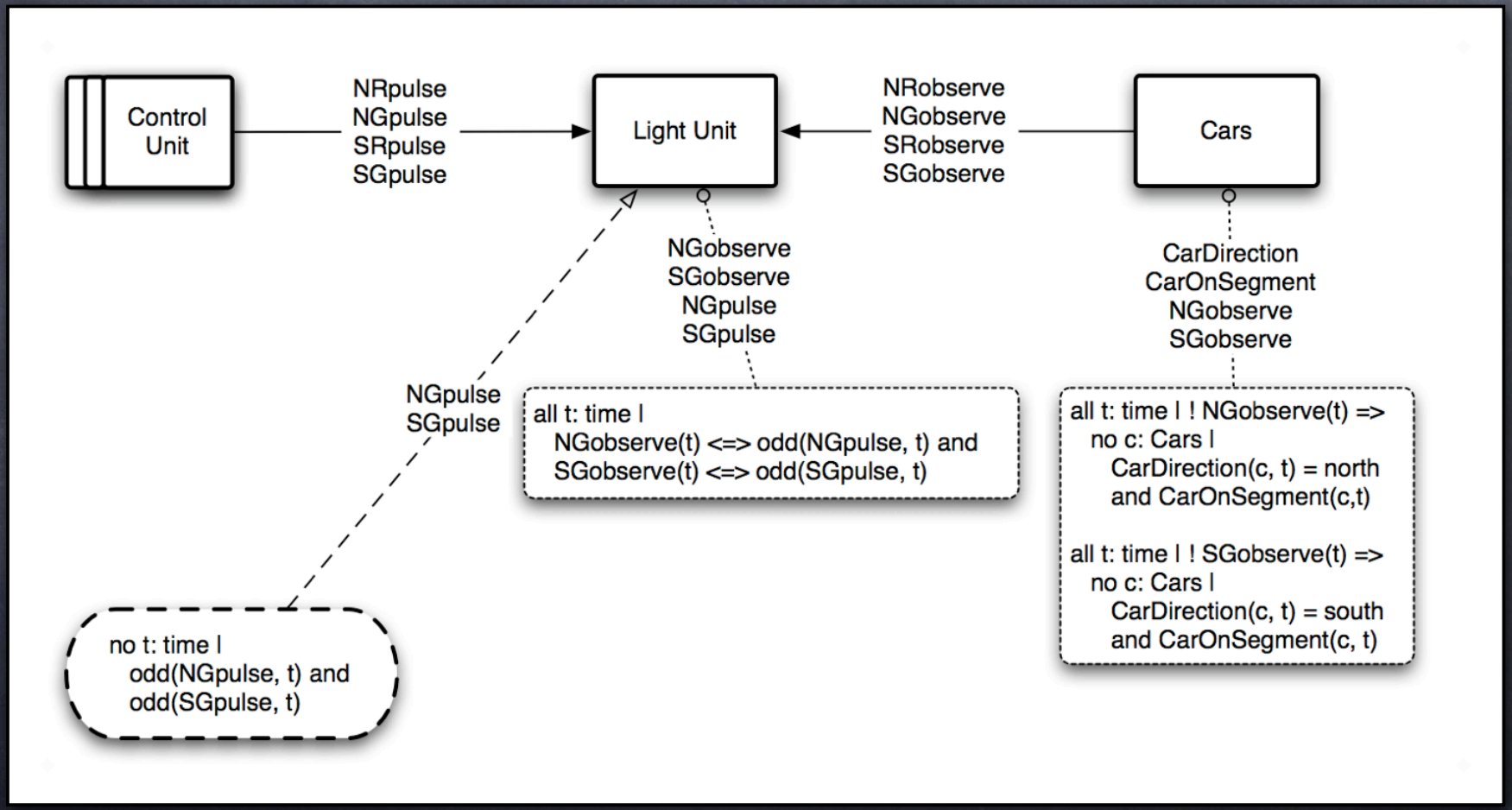
Requirement Progression add domain assumption



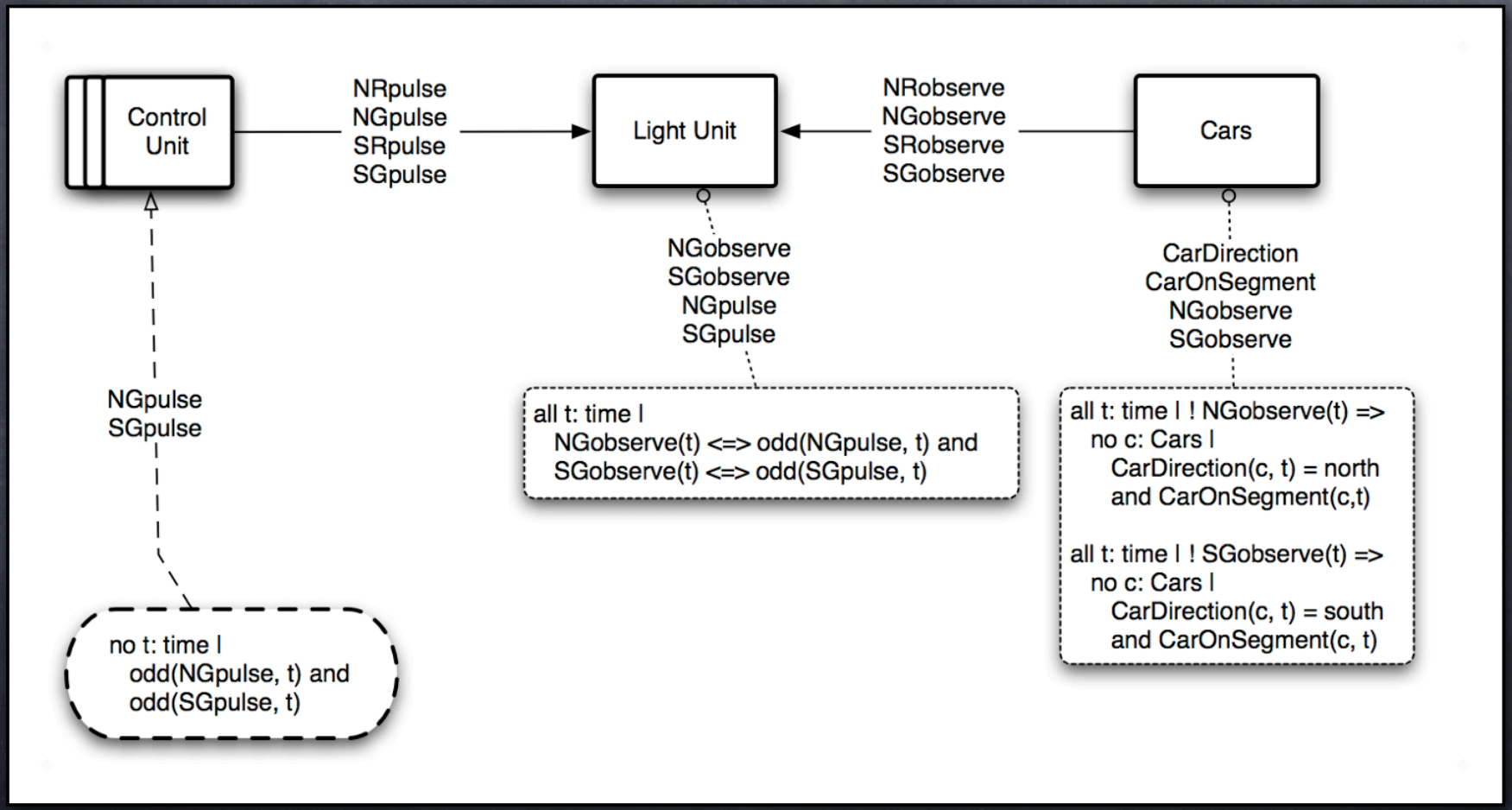
Requirement Progression rephrase requirement



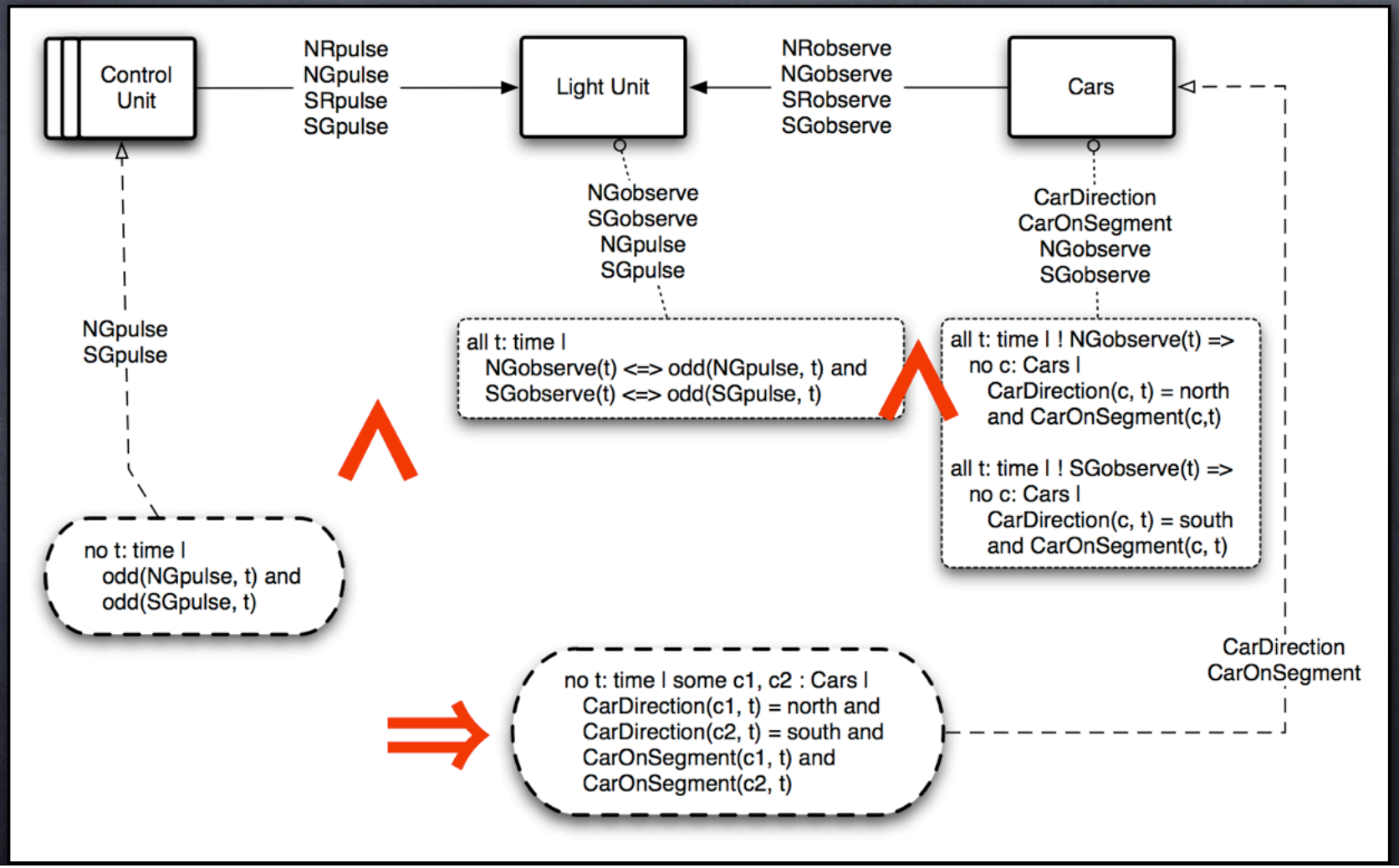
Requirement Progression implication passes



Requirement Progression
push requirement



Requirement Progression end-to-end guarantee

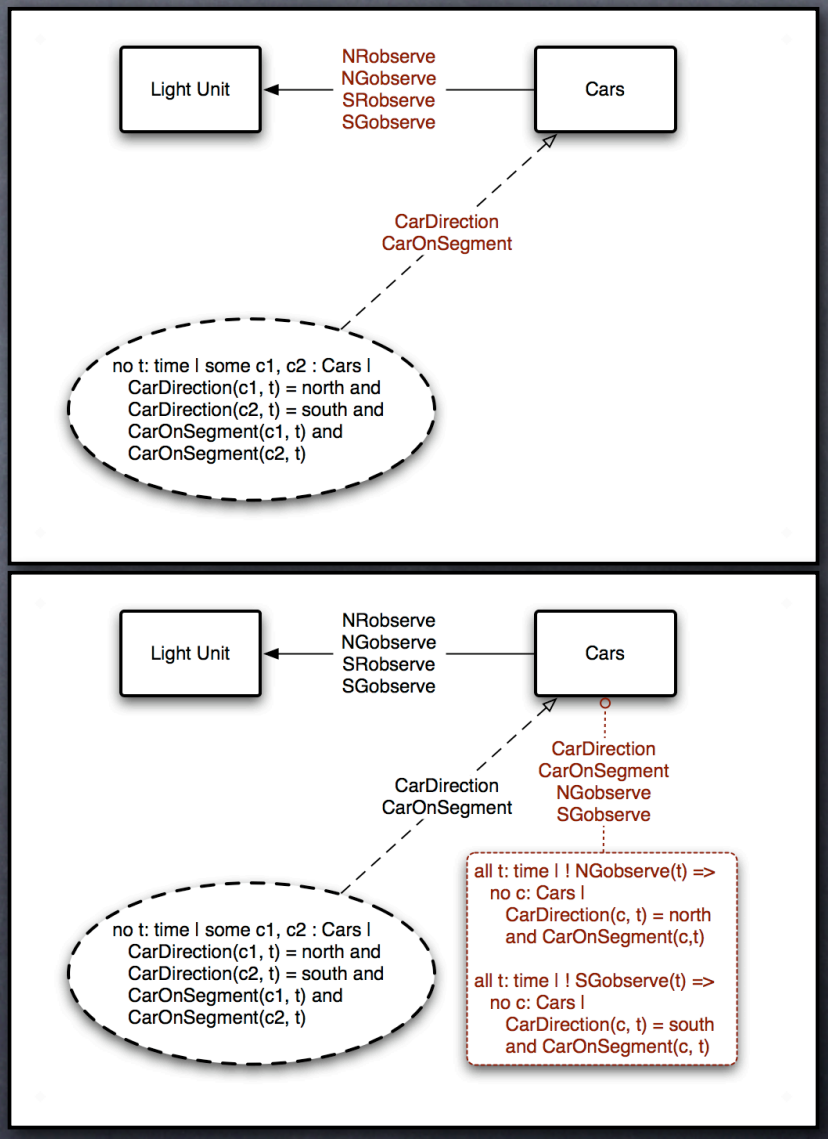


Progression Toolkit

- Add Breadcrumb
- Rephrase Requirement
- Push Arc
- Split/Merge Arc
- Heuristics

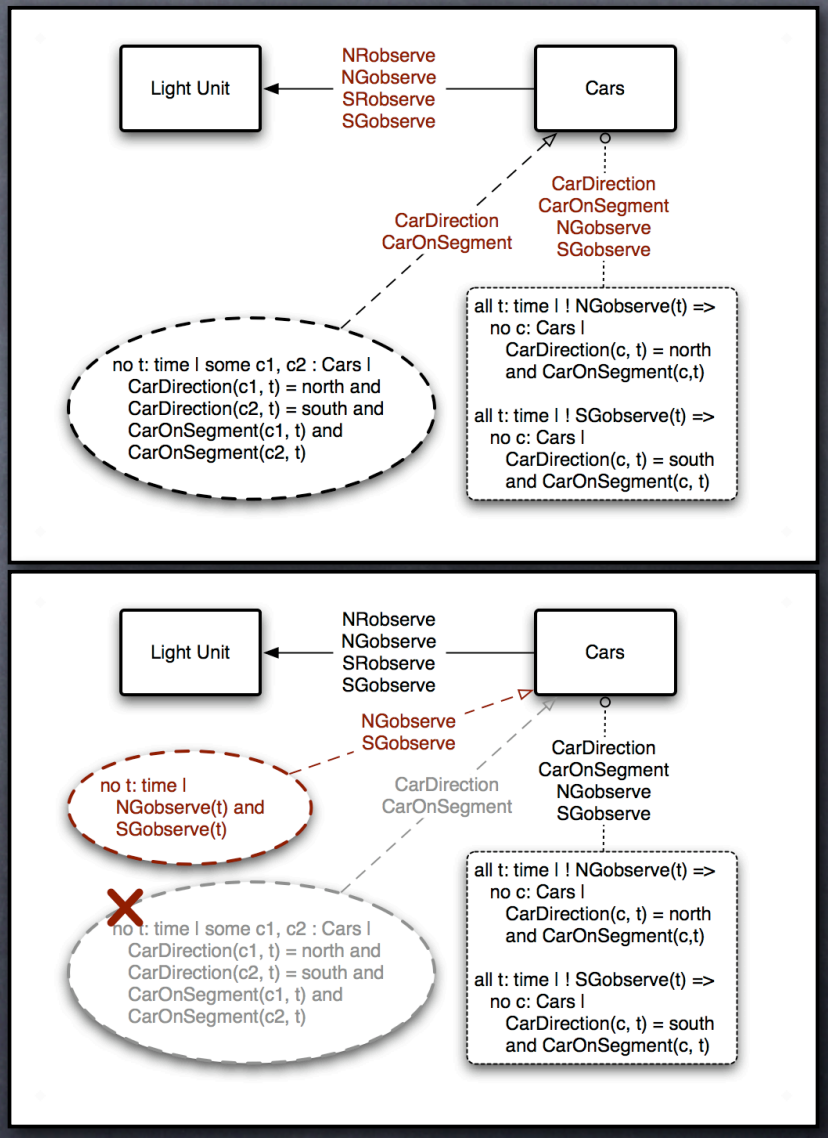
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
- Push Arc
- Split/Merge Arc
- Heuristics



Progression Toolkit

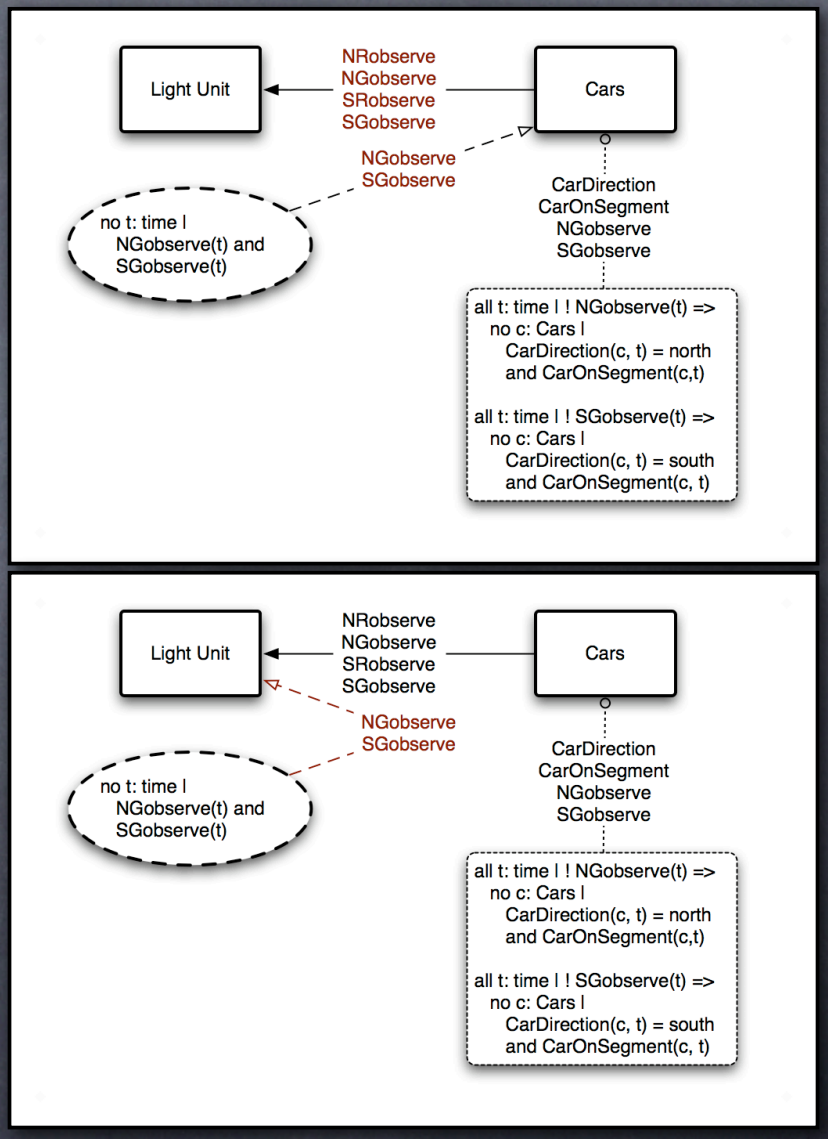
- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
- Split/Merge Arc
- Heuristics



Progression Toolkit

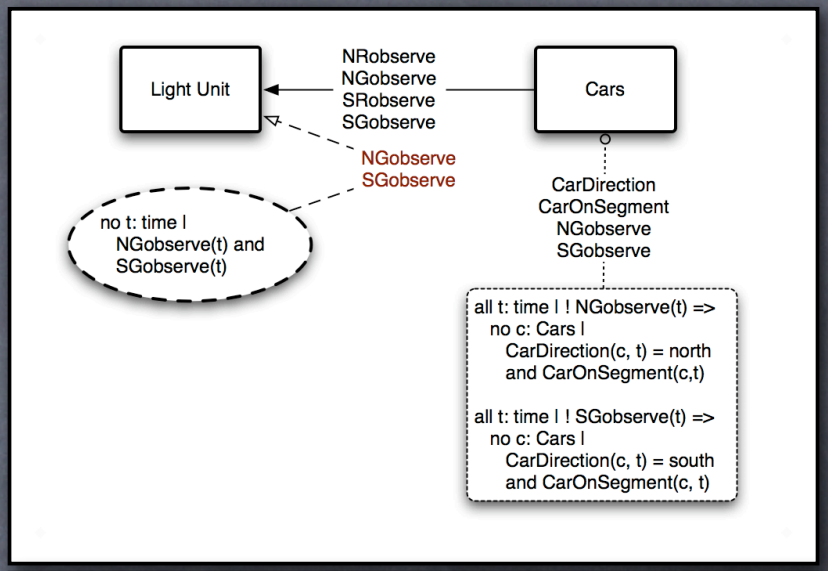
- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc

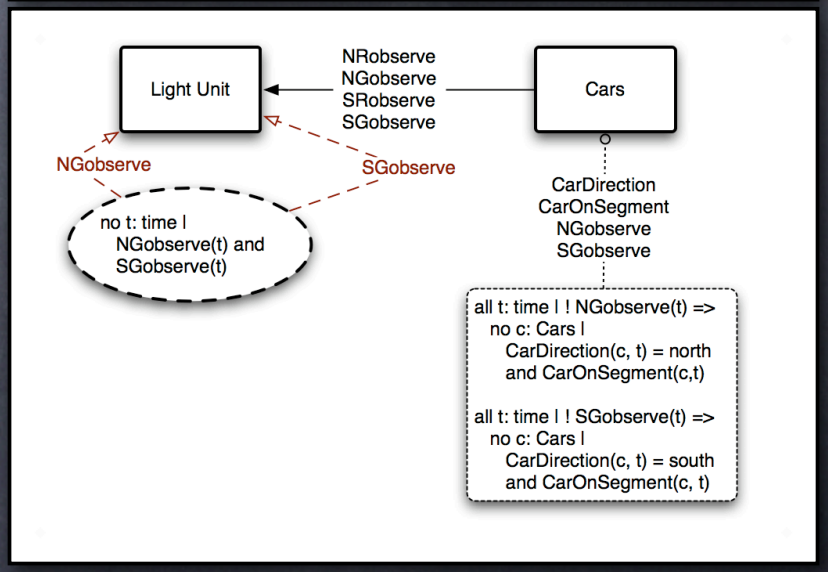
- Heuristics



Progression Toolkit

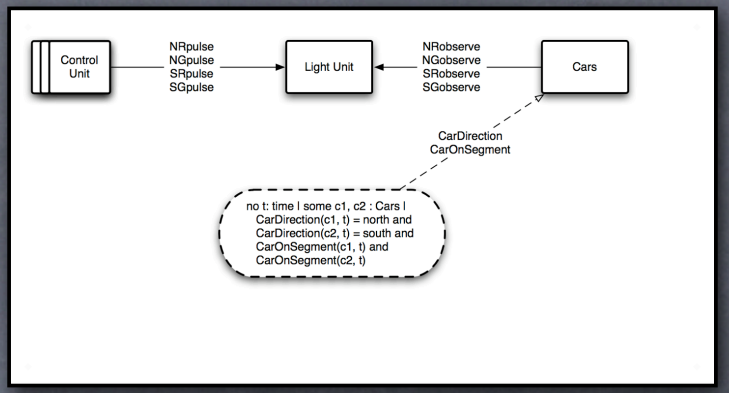
- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics

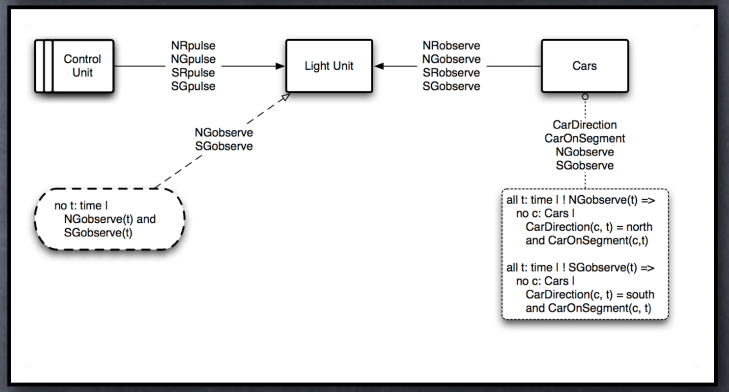


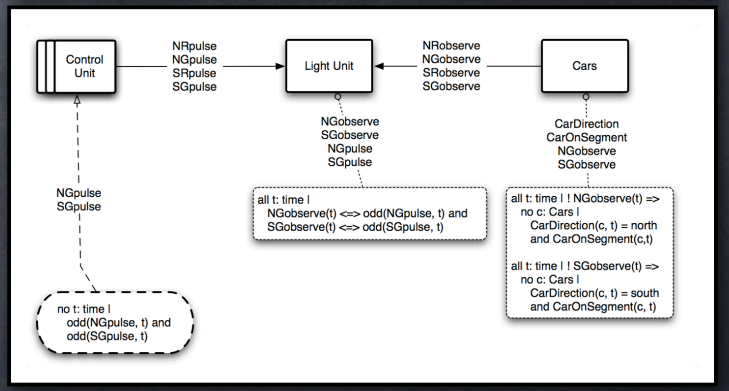


Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  walk req. towards machine
  add / rephrase / push
  guided by informal "frame"
  push multiple arcs separately







Features

guided decomposition

- local reasoning, global guarantee
- result is structured argument

precise language

- unambiguous when dispatched
- automatic checks

explicit (vs implicit) domain assumptions

- "add what you need to make progress"
- tends to produce minimal assumptions
- focuses auditing and analysis
- reduces implementation bias

Electronic Voting

Prêt à Voter

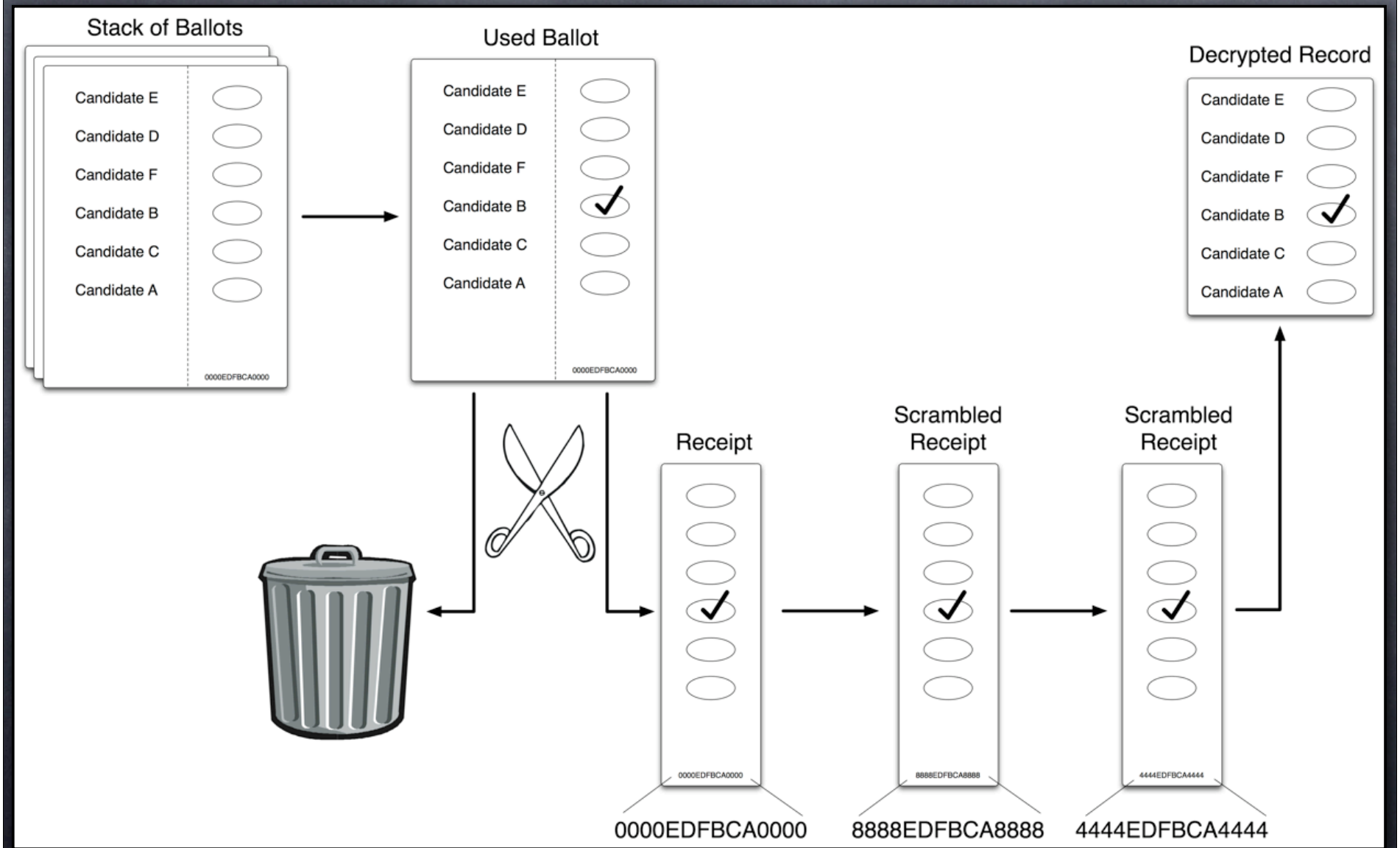
Newcastle University

UNIVERSITY OF SURREY

Peter Ryan

http://www.pretavoter.com/

Prêt à Voter Overview



Voting Requirements

fidelity: count votes correctly

secrecy: protect anonymity

auditability: public can check fidelity

Voting Requirements

fidelity: count votes correctly

- design analysis
- requirement progression

secrecy: protect anonymity

- knowledge/inference model
- leverage fidelity argument

auditability: public can check fidelity

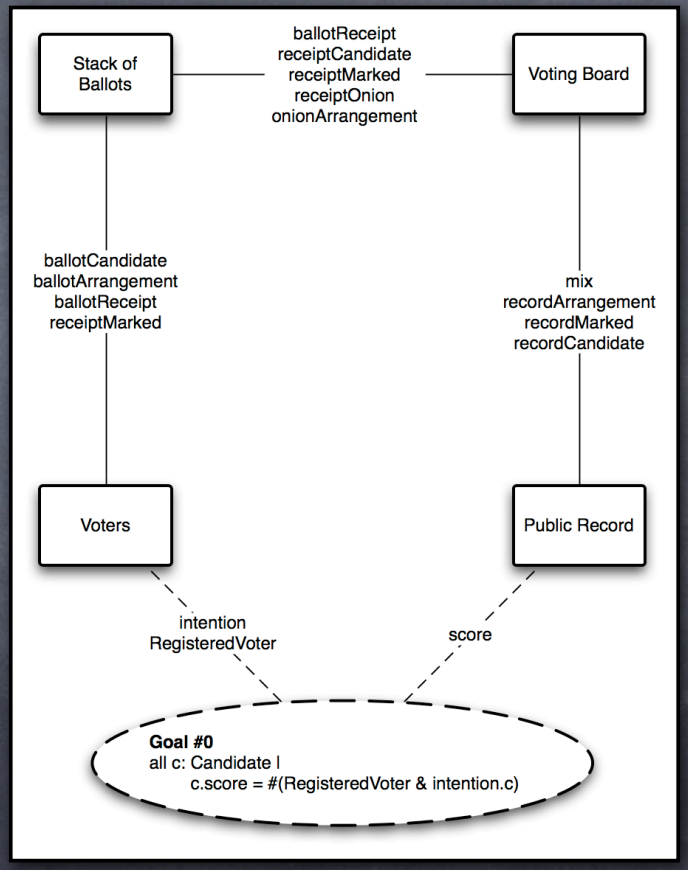
- statistical audits
- but which parts should be audited?

Electronic Voting

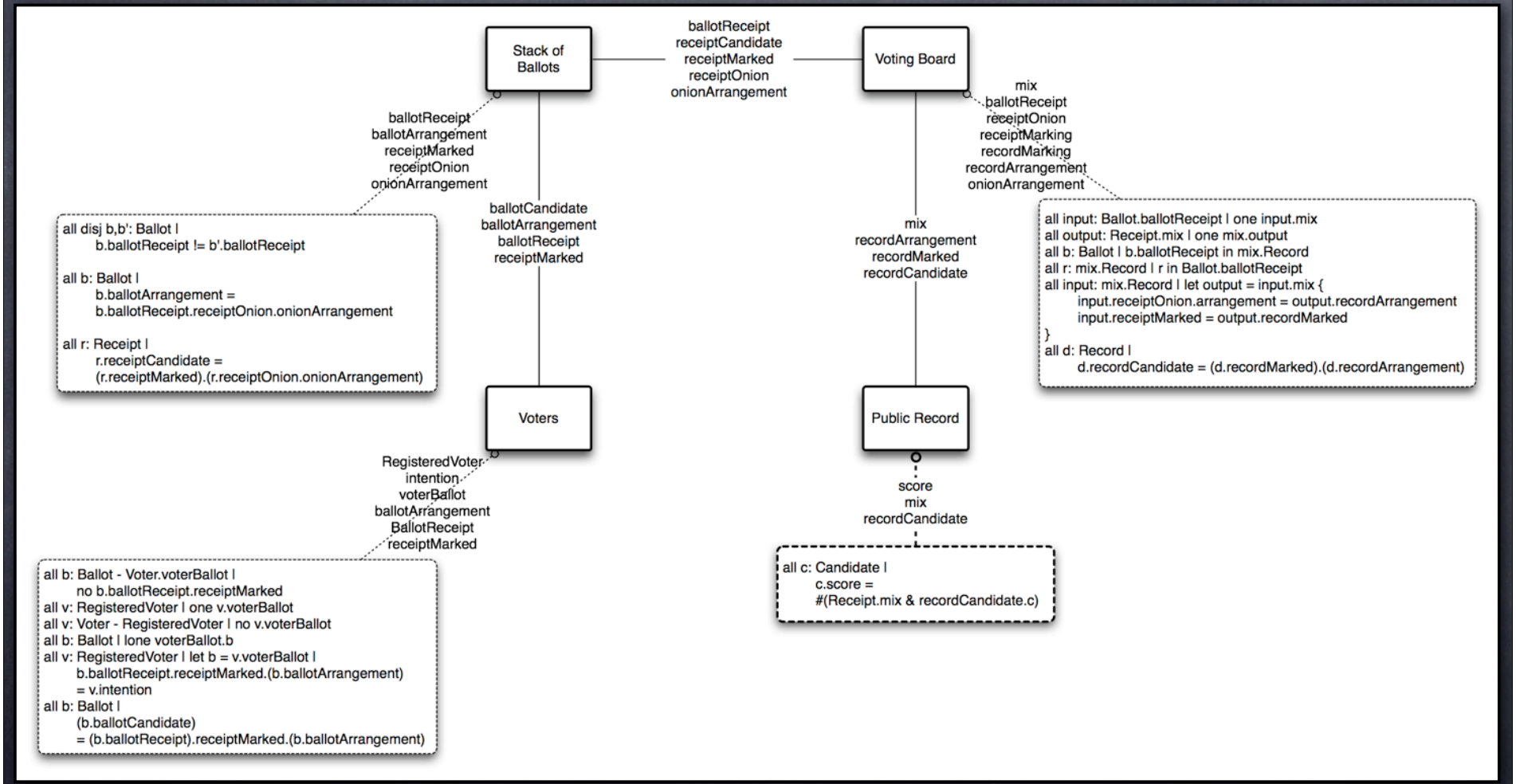
Prêt à Voter

- fidelity
- secrecy
- auditability

Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
Voters
Public Record
intention
RegisteredVoter
score
Goal #0
all c: Candidate |
c.score = #(RegisteredVoter & intention.c)

Fidelity

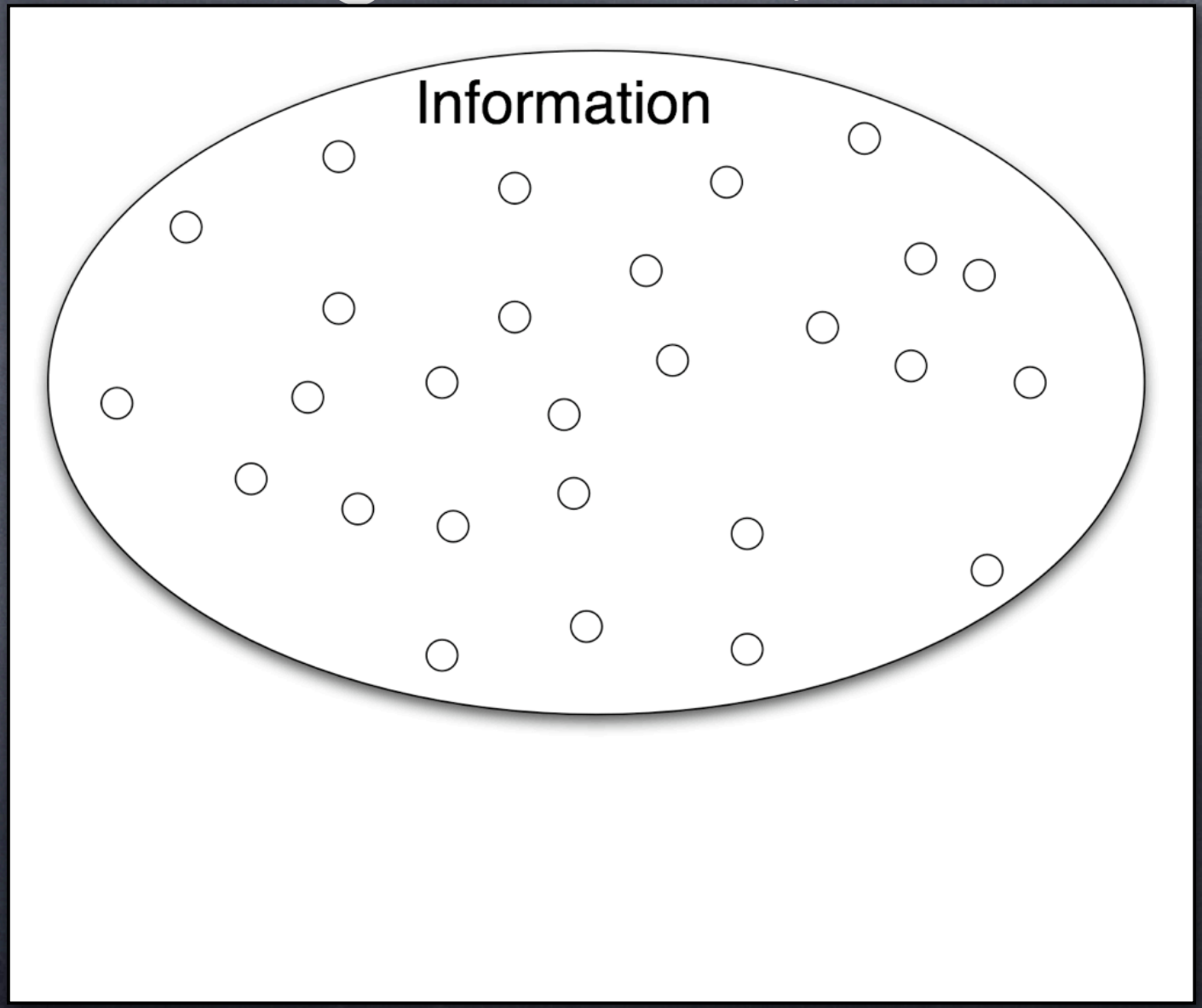

Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotReceipt
ballotArrangement
receiptMarked
receiptOnion
onionArrangement
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
ballotReceipt
receiptOnion
receiptMarking
recordMarking
recordArrangement
onionArrangement
mix
recordArrangement
recordMarked
recordCandidate
all disj b,b': Ballot |
b.ballotReceipt != b'.ballotReceipt
all b: Ballot |
b.ballotArrangement =
b.ballotReceipt.receiptOnion.onionArrangement
all r: Receipt |
r.receiptCandidate =
(r.receiptMarked).(r.receiptOnion.onionArrangement)
all input: Ballot.ballotReceipt | one input.mix
all output: Receipt.mix | one mix.output
all b: Ballot | b.ballotReceipt in mix.Record
all r: mix.Record | r in Ballot.ballotReceipt
all input: mix.Record | let output = input.mix {
input.receiptOnion.arrangement = output.recordArrangement
input.receiptMarked = output.recordMarked
}
all d: Record |
d.recordCandidate = (d.recordMarked).(d.recordArrangement)
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
score
mix
recordCandidate
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
all c: Candidate |
c.score =
#(Receipt.mix & recordCandidate.c)

Electronic Voting

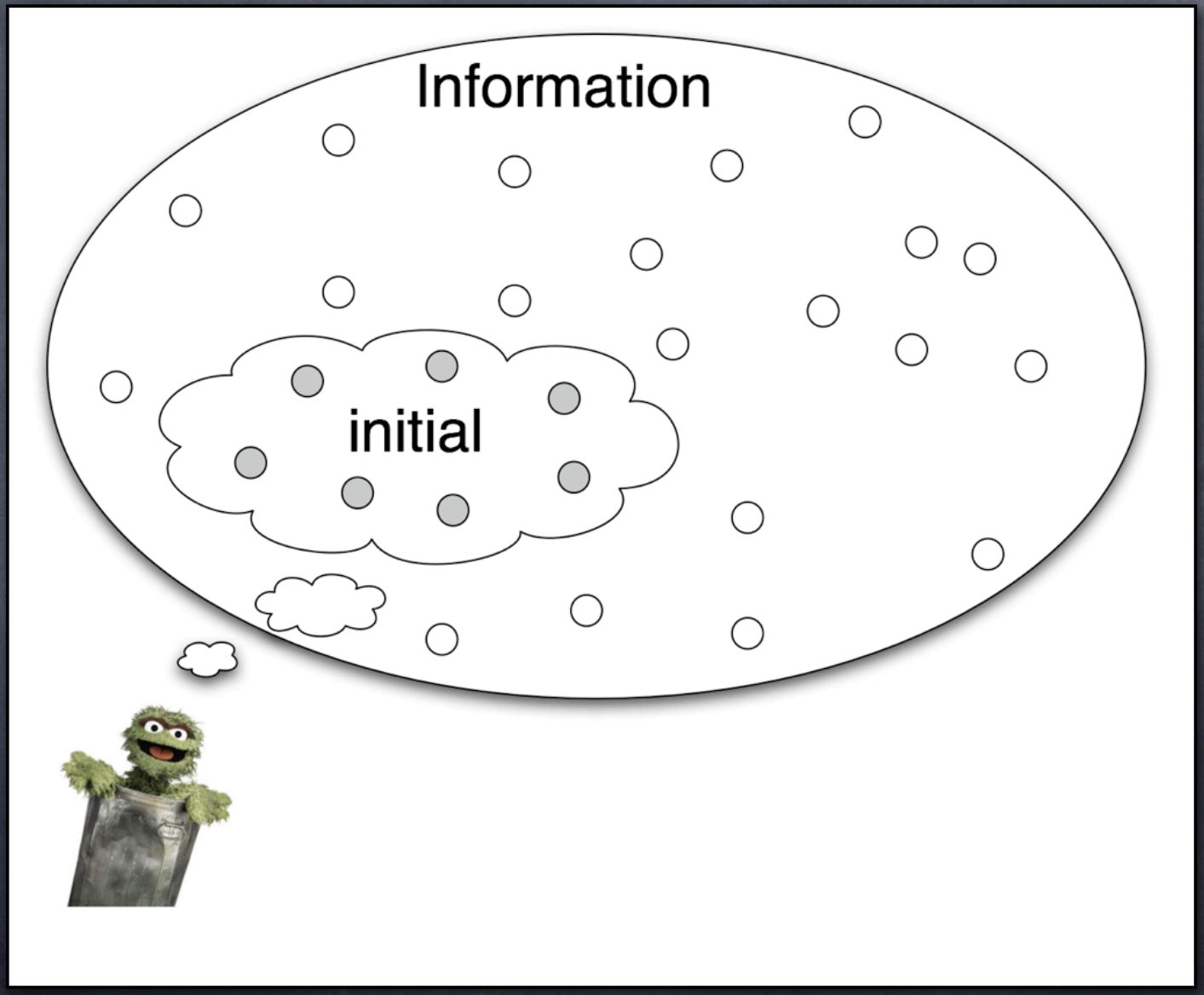
Prêt à Voter

- ✓ fidelity
- ☛ secrecy
- auditability

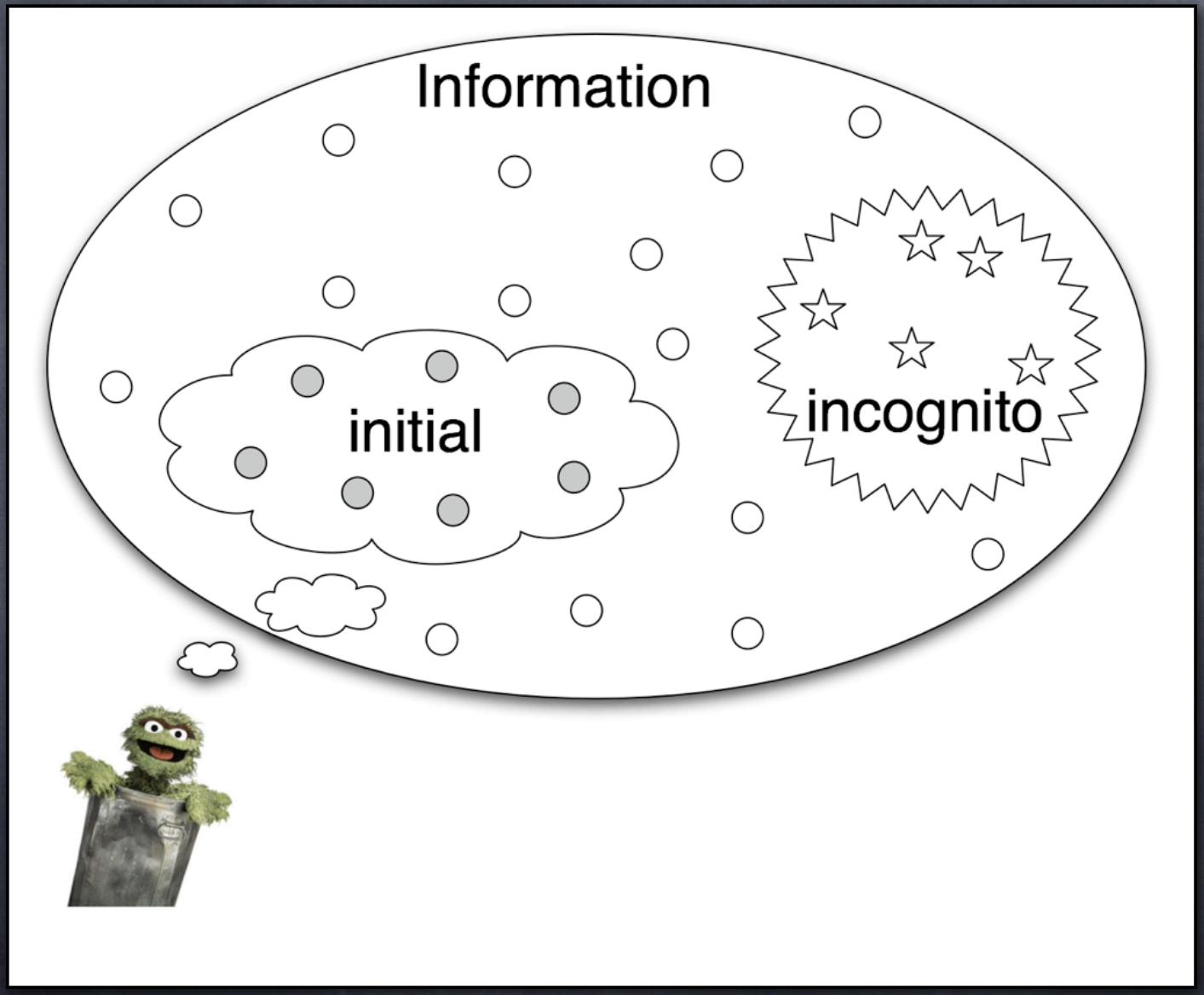
Modeling Secrecy Attacks



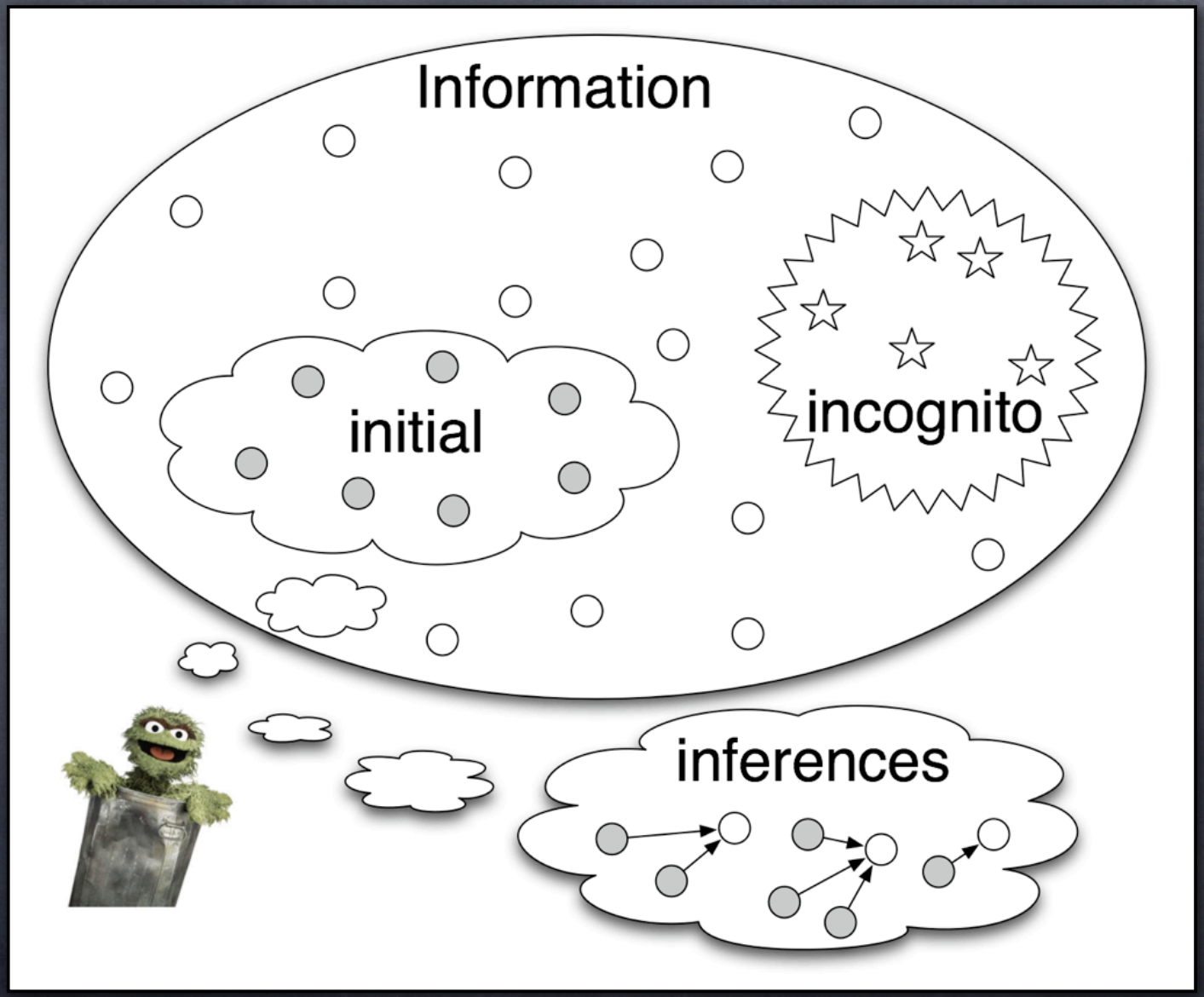
Modeling Secrecy Attacks



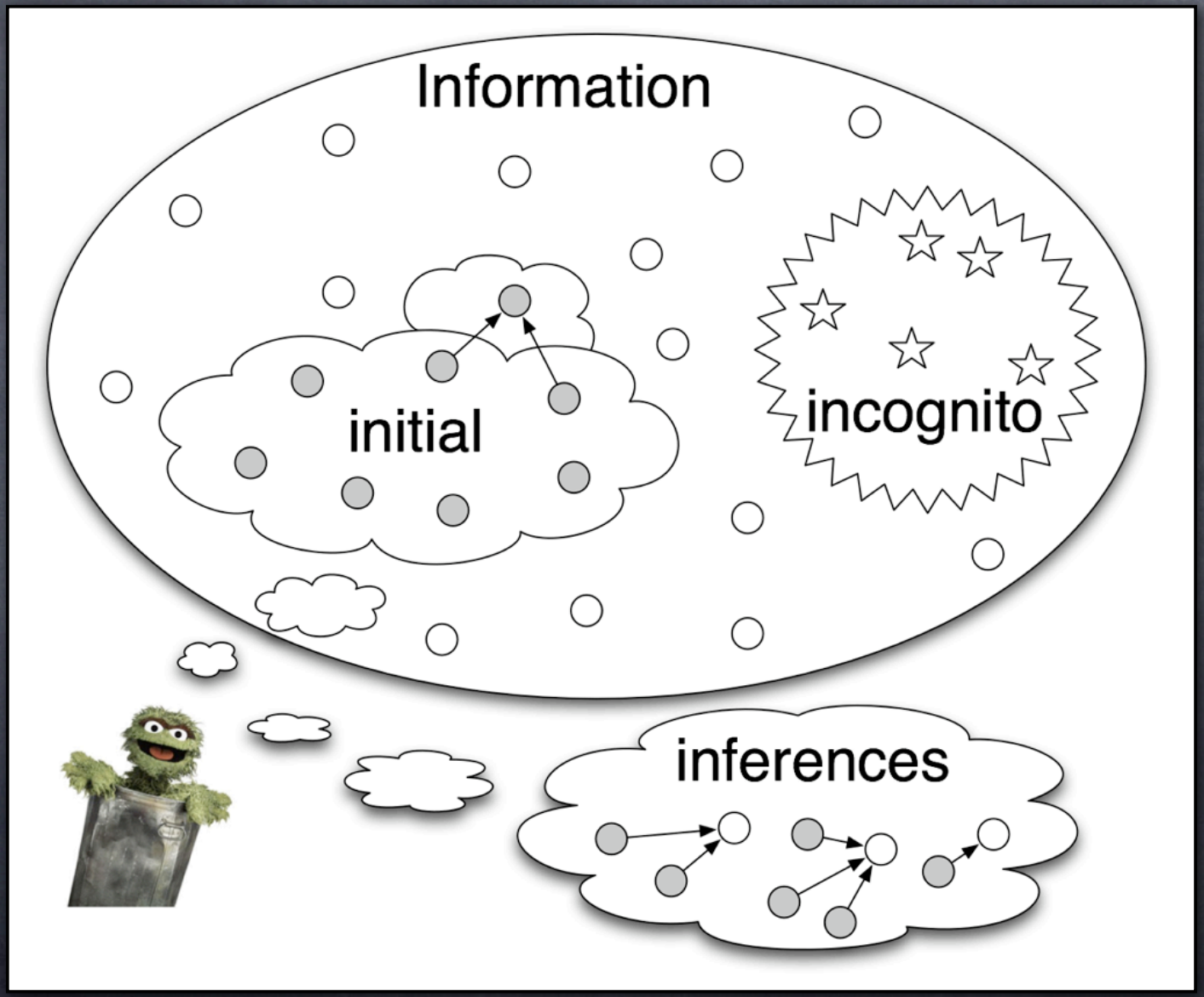
Modeling Secrecy Attacks



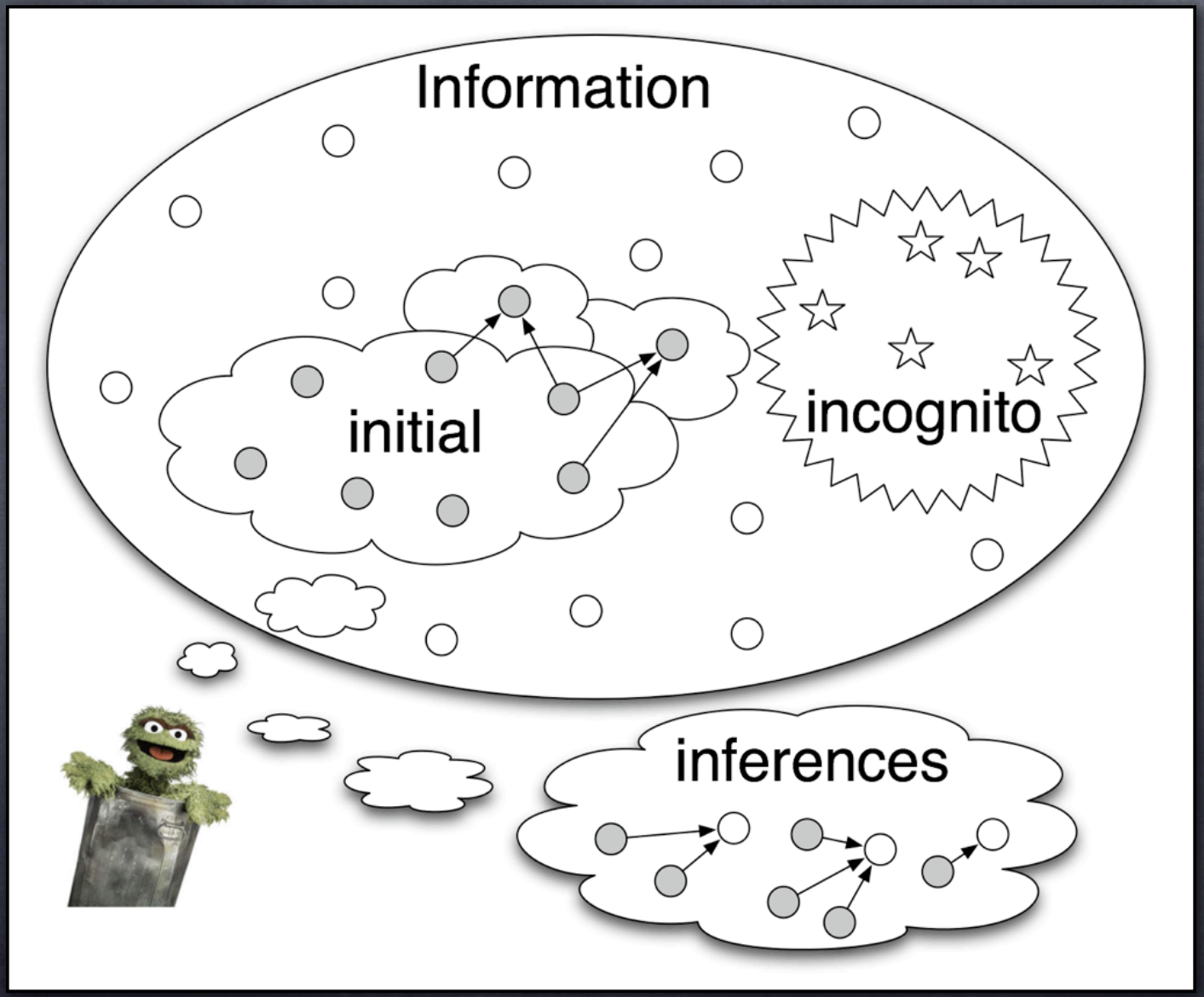
Modeling Secrecy Attacks



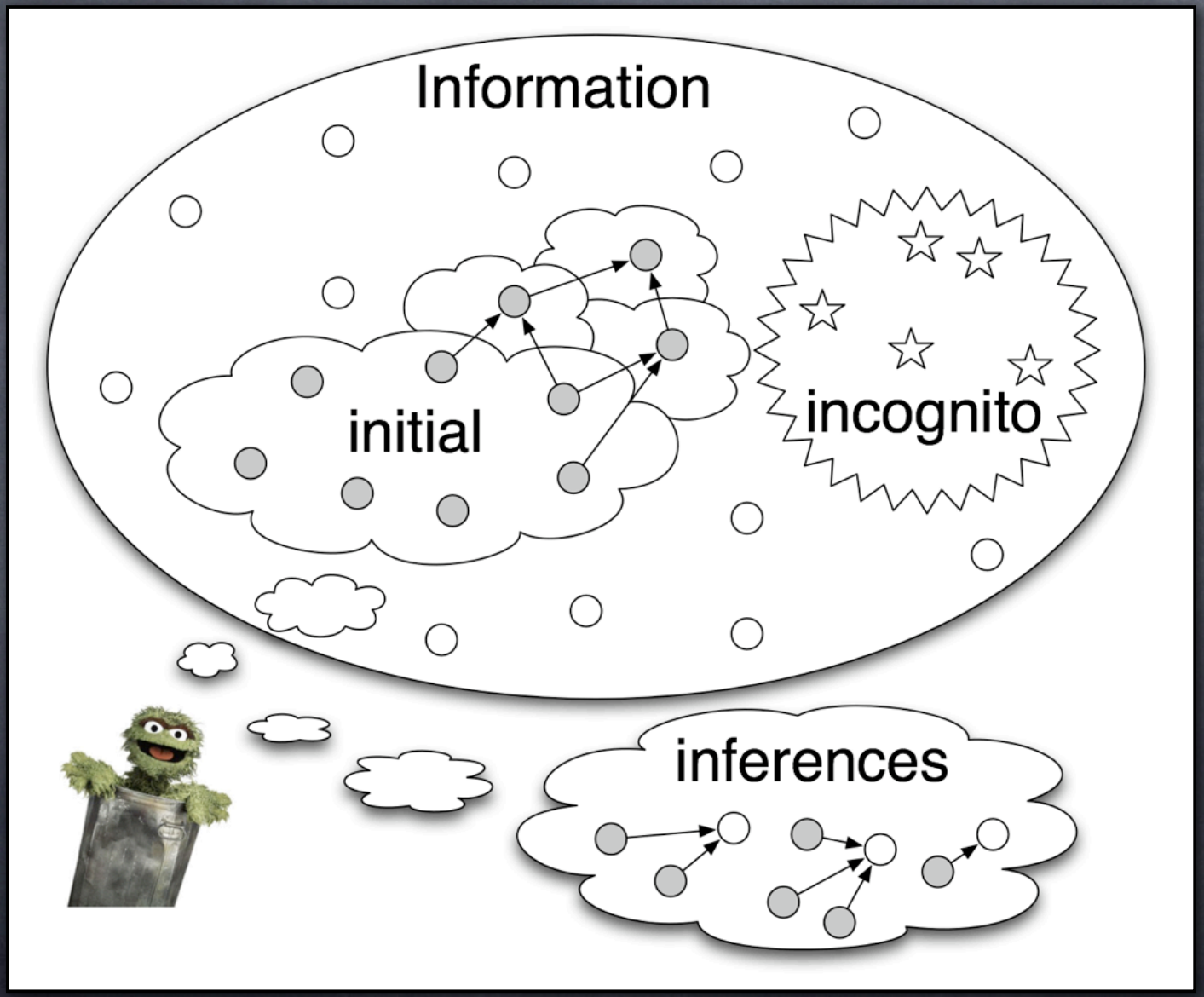
Modeling Secrecy Attacks



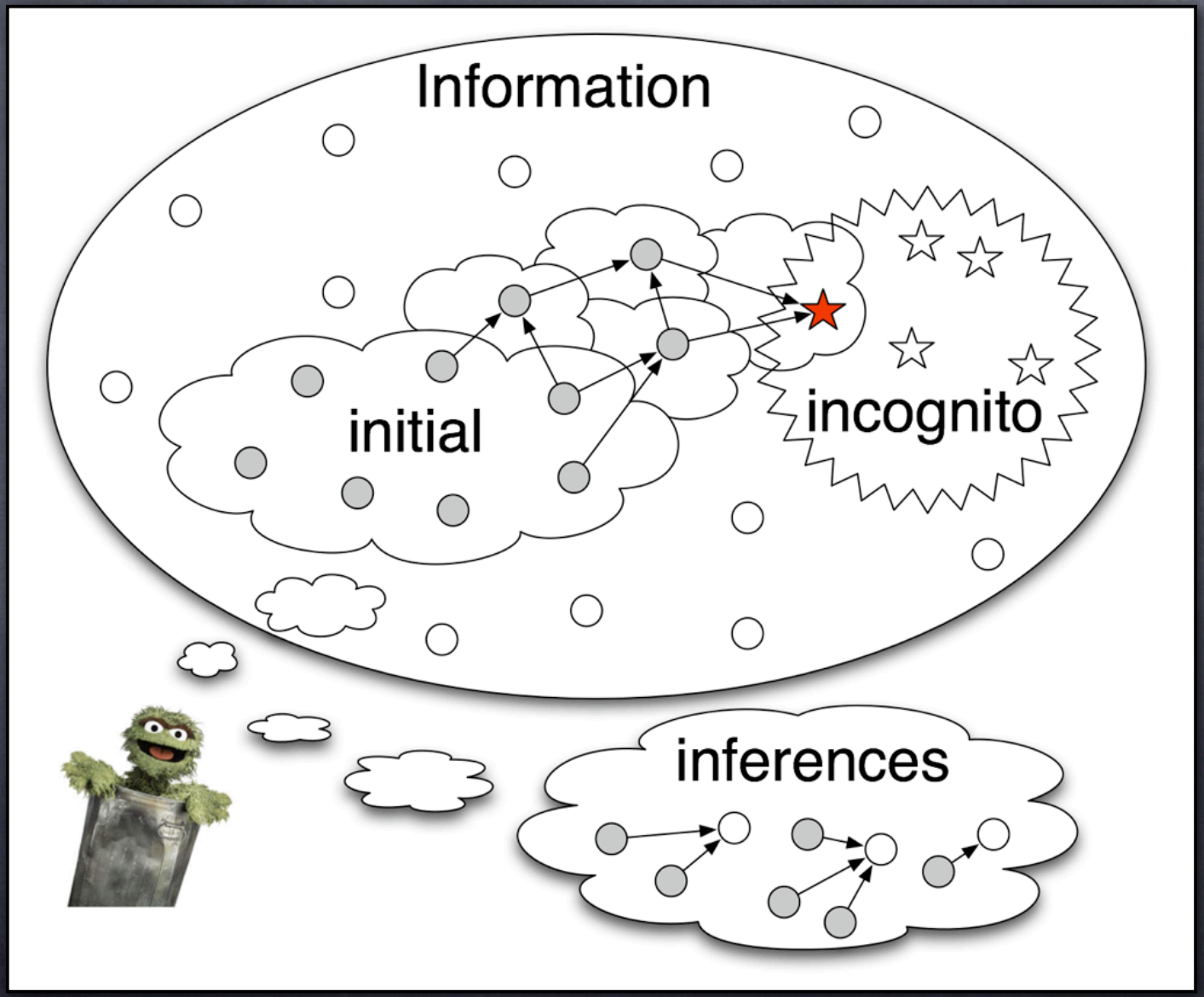
Modeling Secrecy Attacks



Modeling Secrecy Attacks



Modeling Secrecy Attacks



Modeling Secrecy Attacks Information

extend fidelity model

```
sig Voter {
   intention: set Candidate,
   voterBallot: set Ballot,
}

sig Voter {
   intention: set Candidate,
   known_intention: Candidate -> Time,
   voterBallot: set Ballot,
   known_voterBallot: Ballot -> Time,
}
```

Modeling Secrecy Attacks

Initial Data

write constraints on first time step

```
//no telepathy
no known_intention.first

//tear-off receipt, alternate ballots
no known_ballotArrangement.first

//encrypted onions
no known_onionArrangement.first

//re-encryption steps
no known_mix.first
```

intention, mix,
ballotArrangement,
onionArrangement

Modeling Secrecy Attacks Incognito Data

assert secret data is learned

Successful Attack:
The adversary (eventually) knows what ballot was given to some voter and the candidate indicated by that ballot.

```
some v: Voter |
   some v.(known_voterBallot.last)
      .(known_ballotCandidate.last)
```

Modeling Secrecy Attacks Inferences

derive from fidelity assumptions

Breadcrumb
Voters mark their ballots to match their intentions.

```
all v: RegisteredVoter | let b = v.voterBallot |
  b.ballotReceipt.receiptMarked
    .(b.ballotArrangement) = v.intention
```

Inference (Secrecy)
If you know a voter's intention, then you can infer how that voter's ballot is marked (and vice versa).

Modeling Secrecy Attacks Analysis

```
pred attack [] {
//restrictions on knowledge
   no known_intention.first
   no known_ballotArrangement.first
   no known_onionArrangement.first
   no known_mix.first

//only learn via inferences
   explainAdditions[]

//malicious goal
   some v: Voter |
      some v.(known_voterBallot.last)
         .(known_ballotCandidate.last)
}
run attack for 4
```

Electronic Voting

Prêt à Voter

- ✓ fidelity
- ✓ secrecy
- ☛ auditability

How to get auditability

without secrecy - easy

- reveal full fidelity argument (design)
- reveal full operation (implementation)
- e.g. reveal all votes cast

with secrecy - tricky

- cannot reveal all details of operation
- can perform statistical audits
- what to audit?

How to get auditability

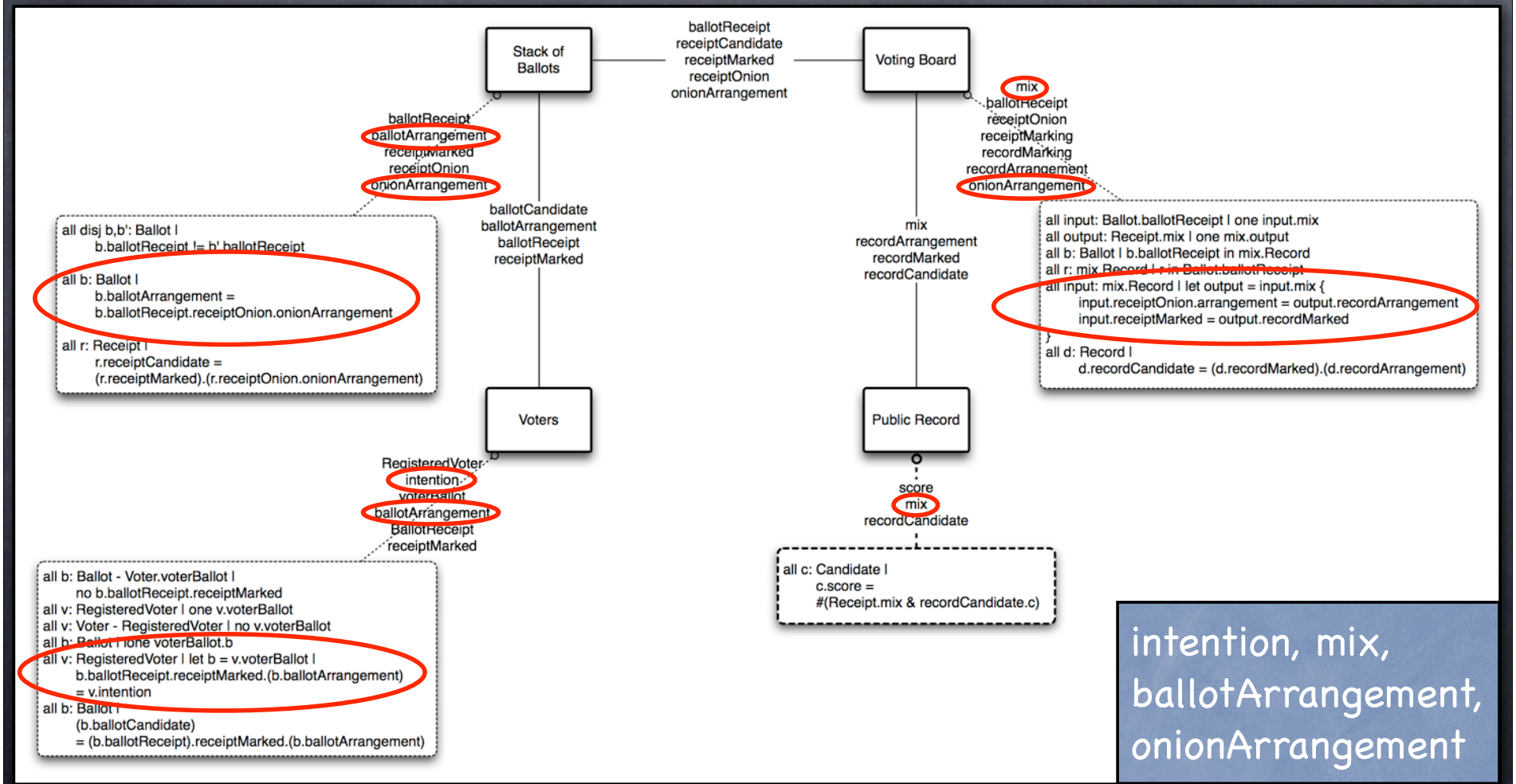
assume fidelity
- breadcrumbs written Alloy model

assume secrecy
- extension of Alloy fidelity model

what should be audited?
- note set of hidden phenomena (secrecy)
- audit referencing assumptions (fidelity)

What to audit?

audit assumptions referencing non-initial information



intention, mix, ballotArrangement, onionArrangement

What to audit?

voters

```
all v: RegisteredVoter | let b = v.voterBallot |
  b.ballotReceipt.receiptMarked.(b.ballotArrangement)
  = v.intention
```

ballots

```
all b: Ballot |
  b.ballotArrangement
  = b.ballotReceipt.receiptOnion.onionArrangement
```

voting board

```
all input: mix.Record | let output = input.mix {
  input.receiptOnion.arrangement
    = output.recordArrangement
  input.receiptMarked = output.recordMarked
}
```

What to audit?

voters

- "voters mark ballots according to intention"
- user study on comprehension, execution

ballots

```
all b: Ballot |
  b.ballotArrangement
  = b.ballotReceipt.receiptOnion.onionArrangement
```

voting board

```
all input: mix.Record | let output = input.mix {
  input.receiptOnion.arrangement
    = output.recordArrangement
  input.receiptMarked = output.recordMarked
}
```

What to audit?

voters

- "voters mark ballots according to intention"
- user study on comprehension, execution

ballots

- "onion arrangements match their ballots"
- randomly sample, discard

voting board

```
all input: mix.Record | let output = input.mix {
  input.receiptOnion.arrangement
    = output.recordArrangement
  input.receiptMarked = output.recordMarked
}
```

What to audit?

voters

- "voters mark ballots according to intention"
- user study on comprehension, execution

ballots

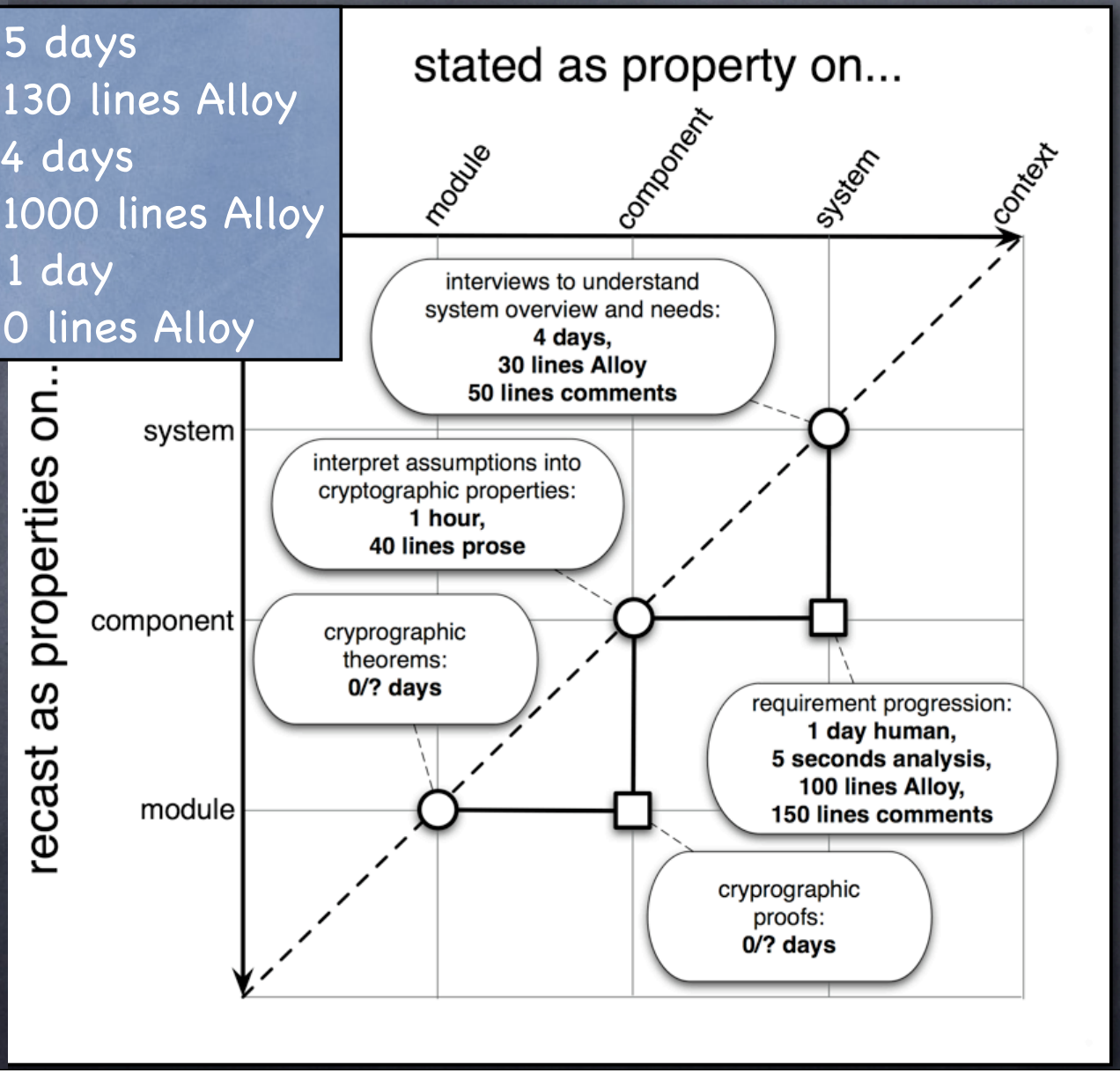
- "onion arrangements match their ballots"
- randomly sample, discard

voting board

- "re-encryption preserves ordering/marking"
- multiple re-encryption steps
- partial random reveal of each step

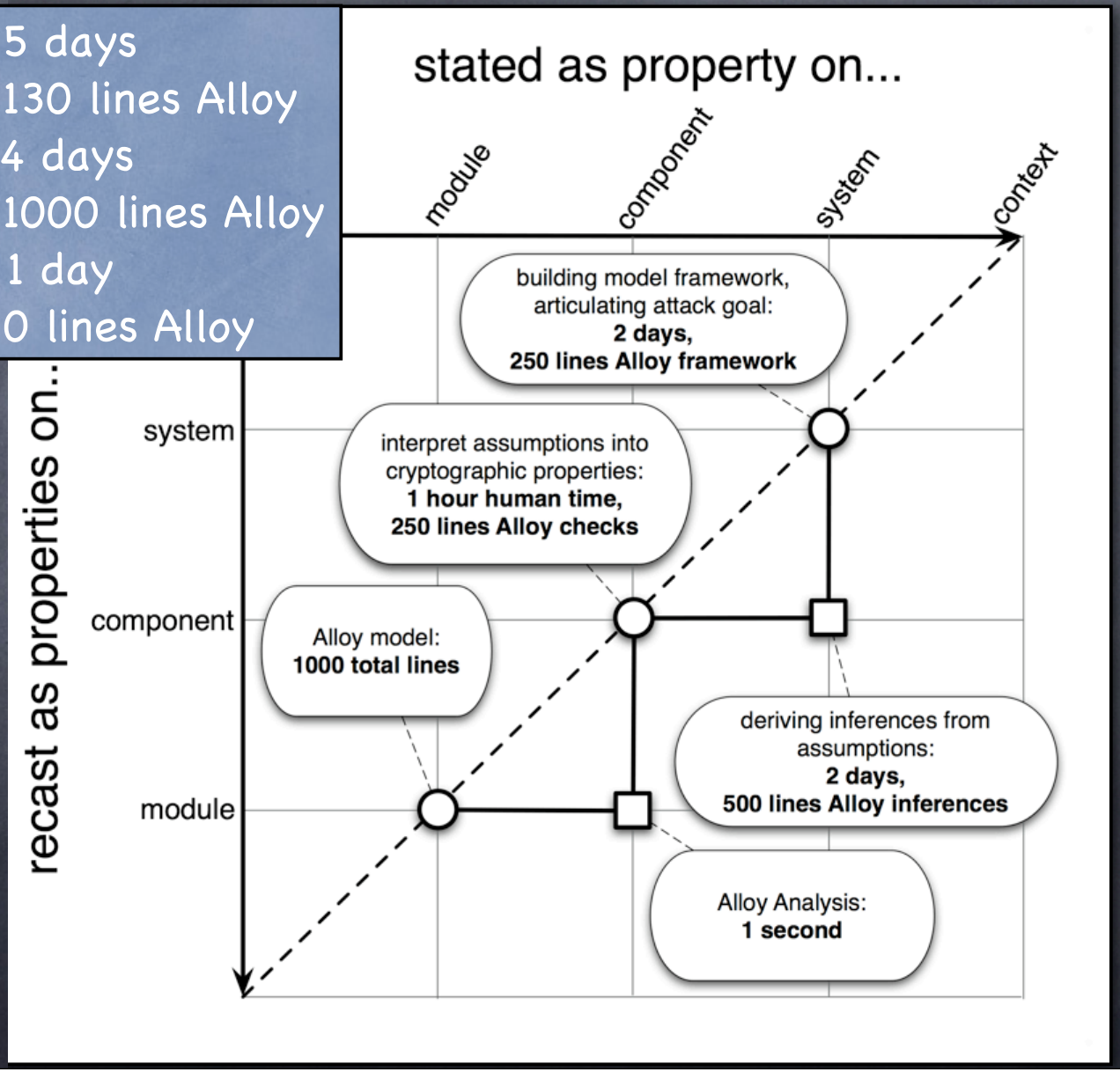
Fidelity Timetable

fidelity:	5 days
	130 lines Alloy
secrecy:	4 days
	1000 lines Alloy
audit list:	1 day
	0 lines Alloy



Secrecy Timetable

fidelity:	5 days
	130 lines Alloy
secrecy:	4 days
	1000 lines Alloy
audit list:	1 day
	0 lines Alloy



Discoveries

dependability argument

- articulate, analyze, document
- separate system/crypto arguments
- confirmed fidelity, secrecy, auditability

build arguments in tandem: easier, thorough

- **fidelity** with req. prog.
- **secrecy** based on fidelity
- **audit list** read off of fidelity/secrecy

discoveries

- no need to absorb marking into onion
- do need alternate ballots in booth

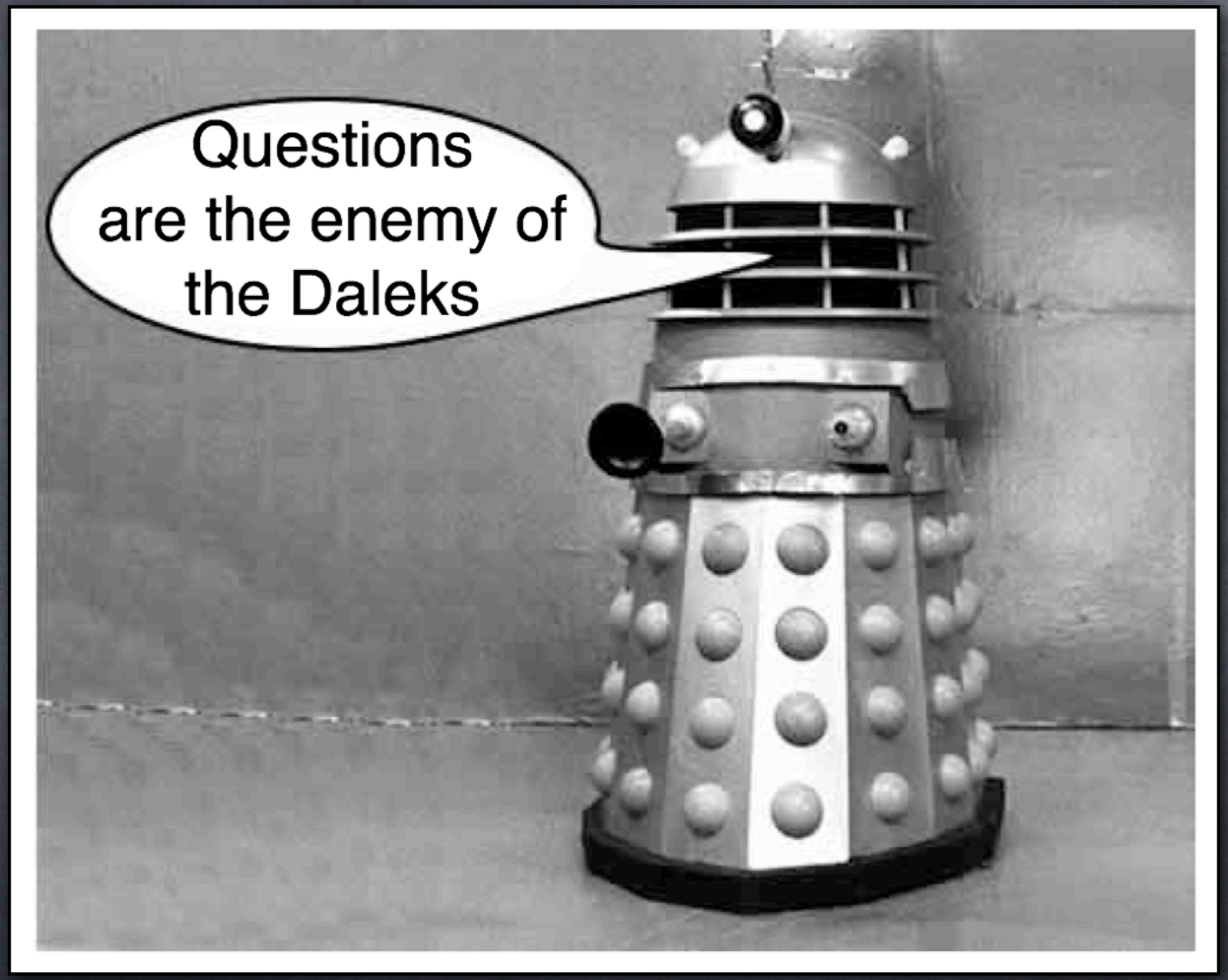
Interests

link high level goals to low level specs

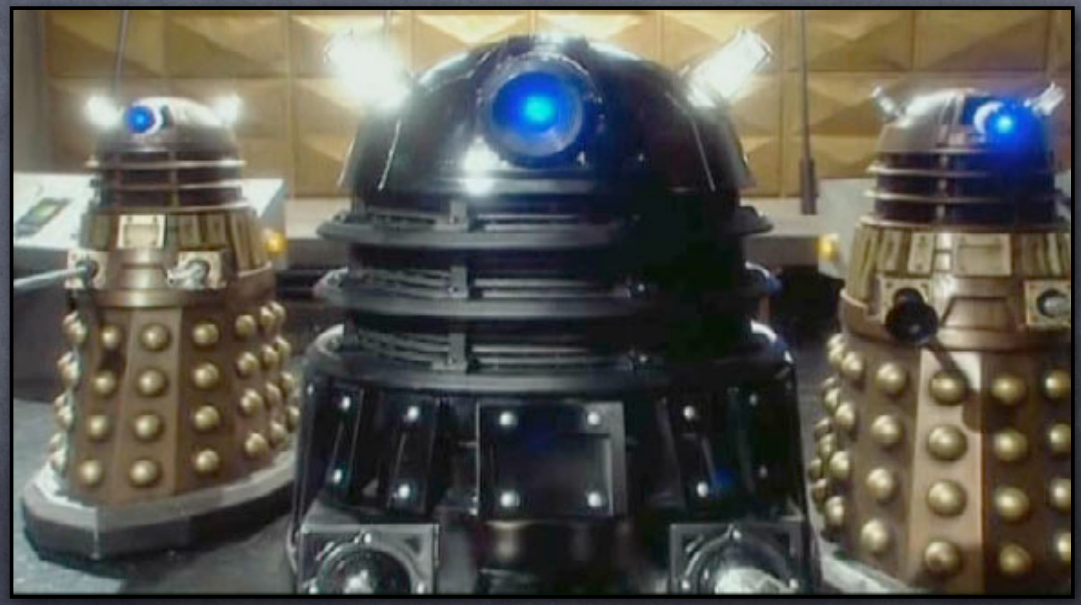
- talk to managers/users/customers
- talk to diff. types of engineers
- help articulate goals, capabilities
- document argument in usable form

interpreting technical info for user

- requirement progression guidance
- show secrecy attack traces
- role of constraint in model
- magic layout for Alloy


Questions
are the enemy of
the Daleks

Supplemental Slides



Providing Dependability

what must a dependability argument do?

justify system requirement
- cross-cutting, not monolithic

justify component assumptions
- hard evidence / clear sign-off

connect requirement to assumptions
- end-to-end argument
- disciplined checked

Why "Dependability"?

avoids undesired connotations

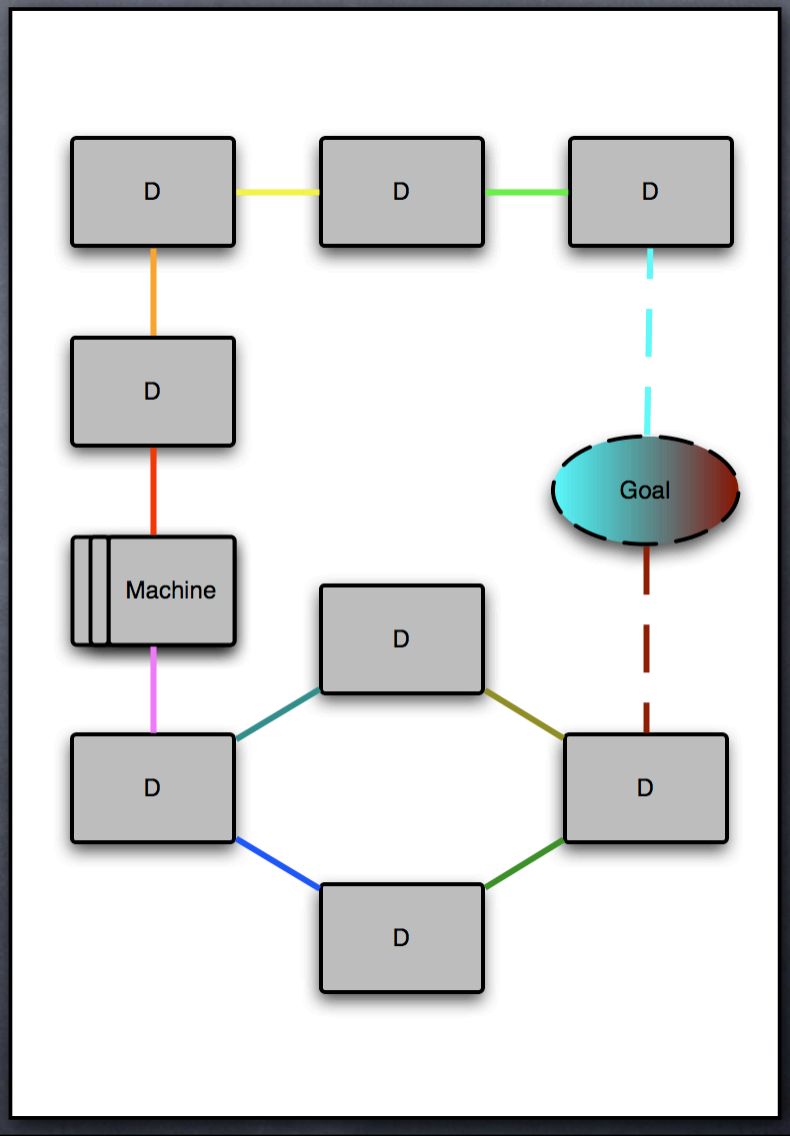
reliability suggests...
- avoiding downtime

safety suggests
- avoiding human risk
- avoiding disaster

correctness suggests...
- absolutes (vs. reasonable confidence)

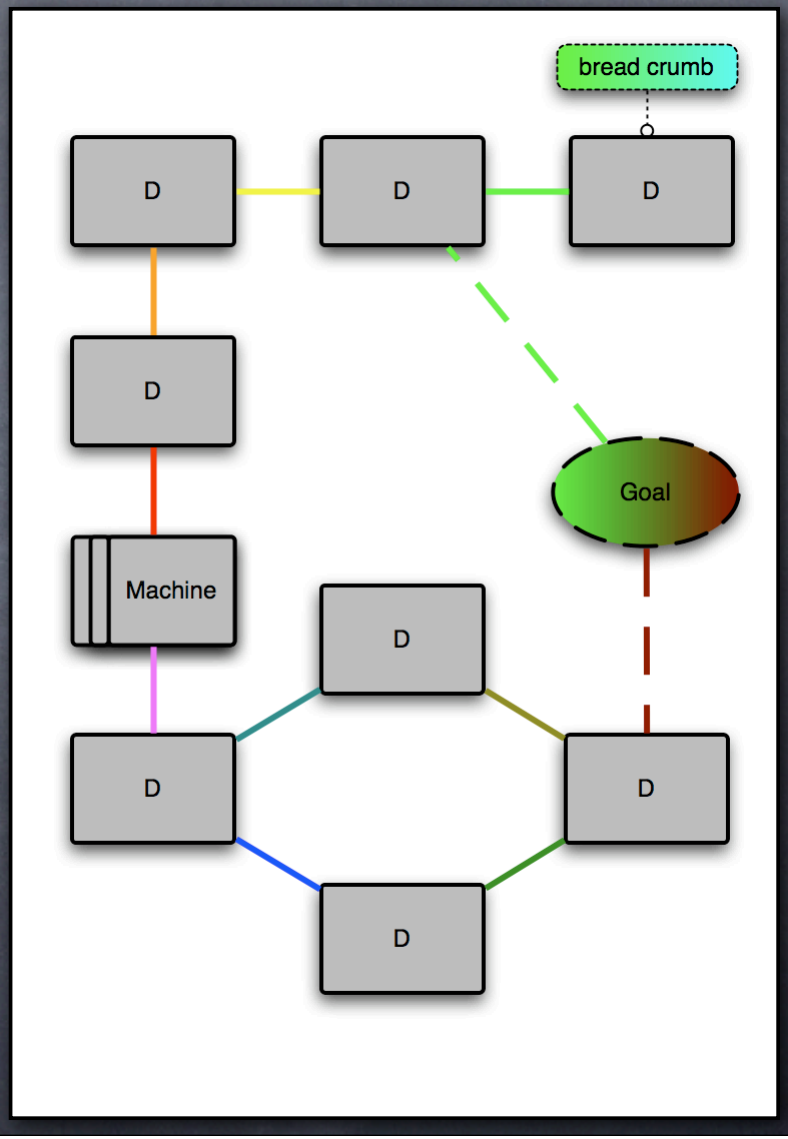
Progression Toolkit

- **Add Breadcrumb**
  on phenomena of 1 domain
- **Rephrase Requirement**
  new ∧ breadcrumb ⇒ old
- **Push Arc**
  if phenomena are shared
- **Split/Merge Arc**
  nothing else changes
- **Heuristics**
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



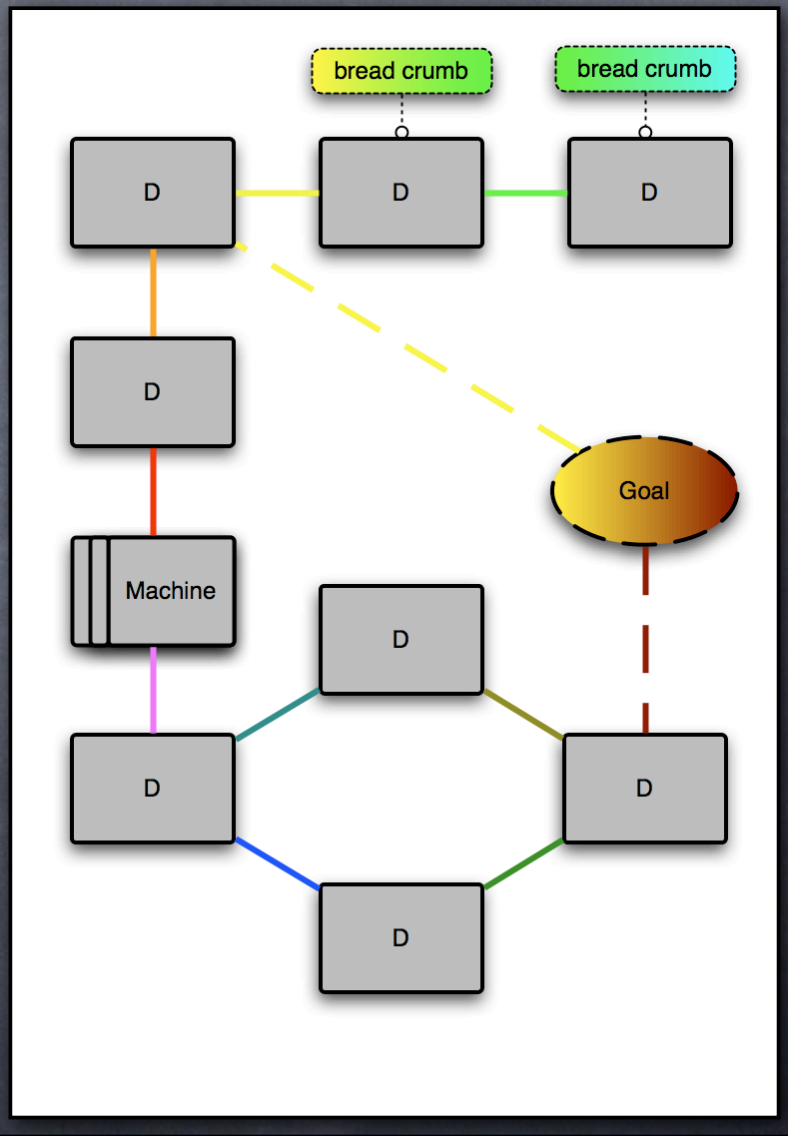
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



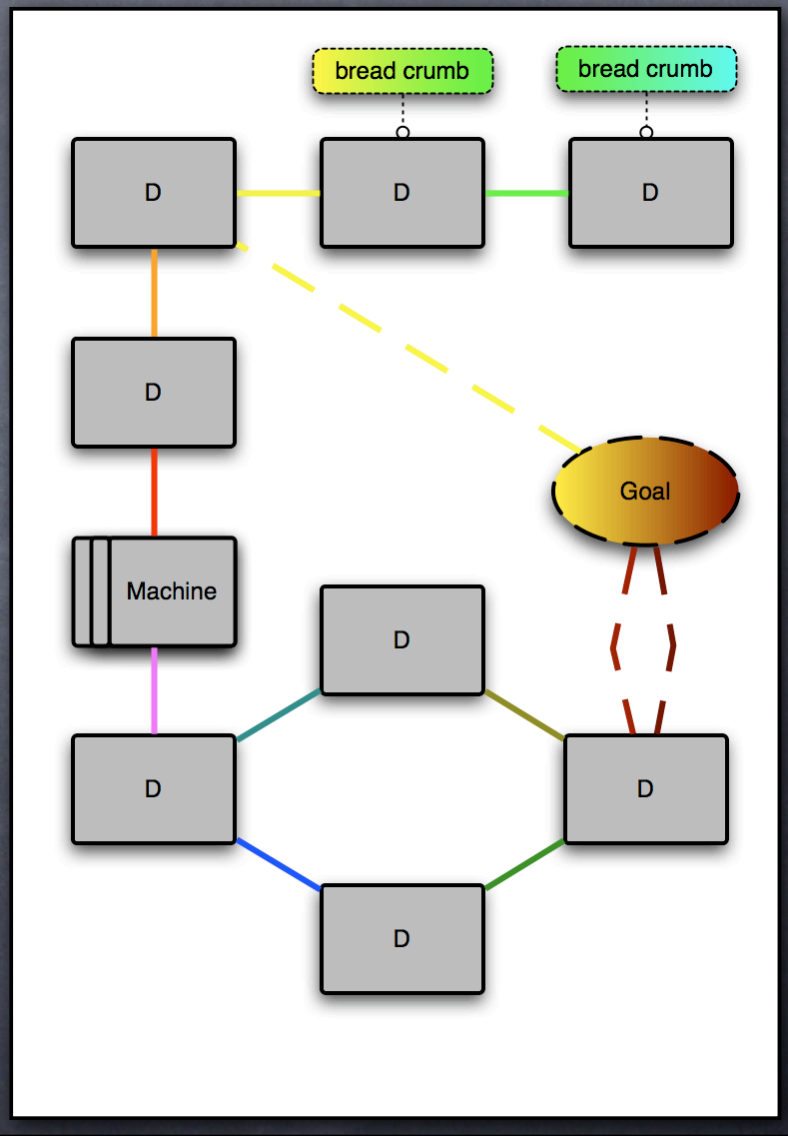
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



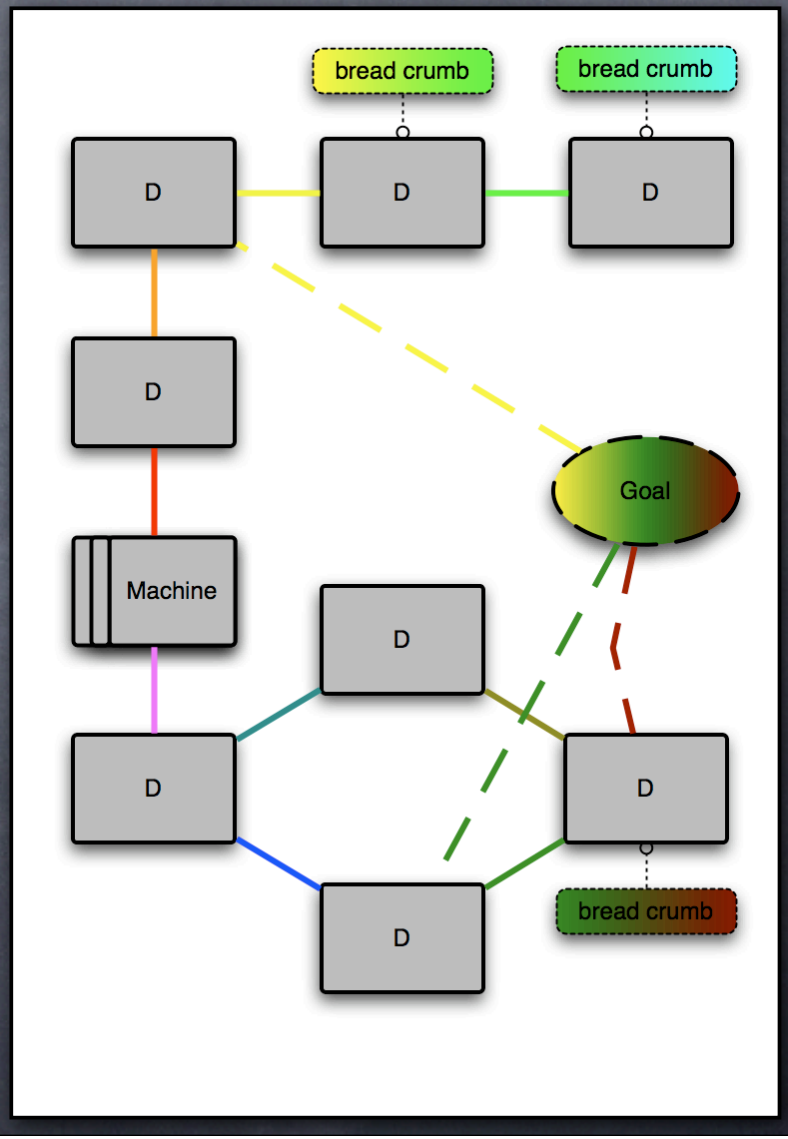
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



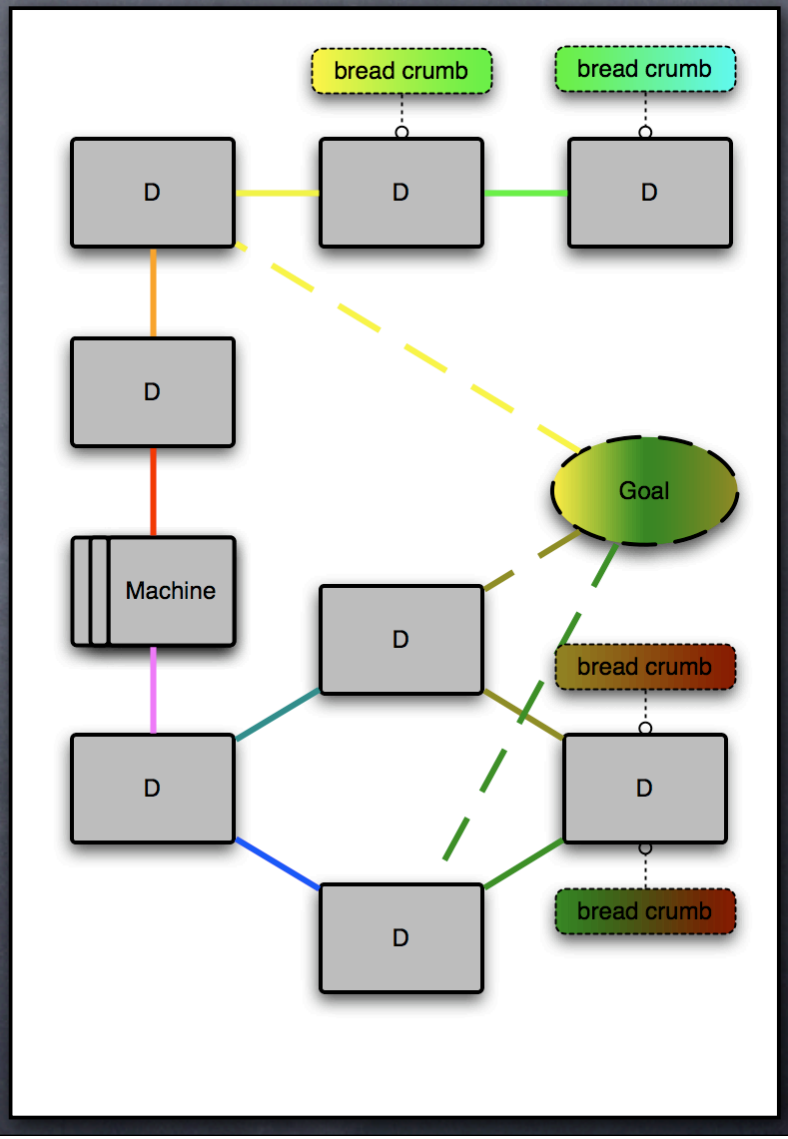
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



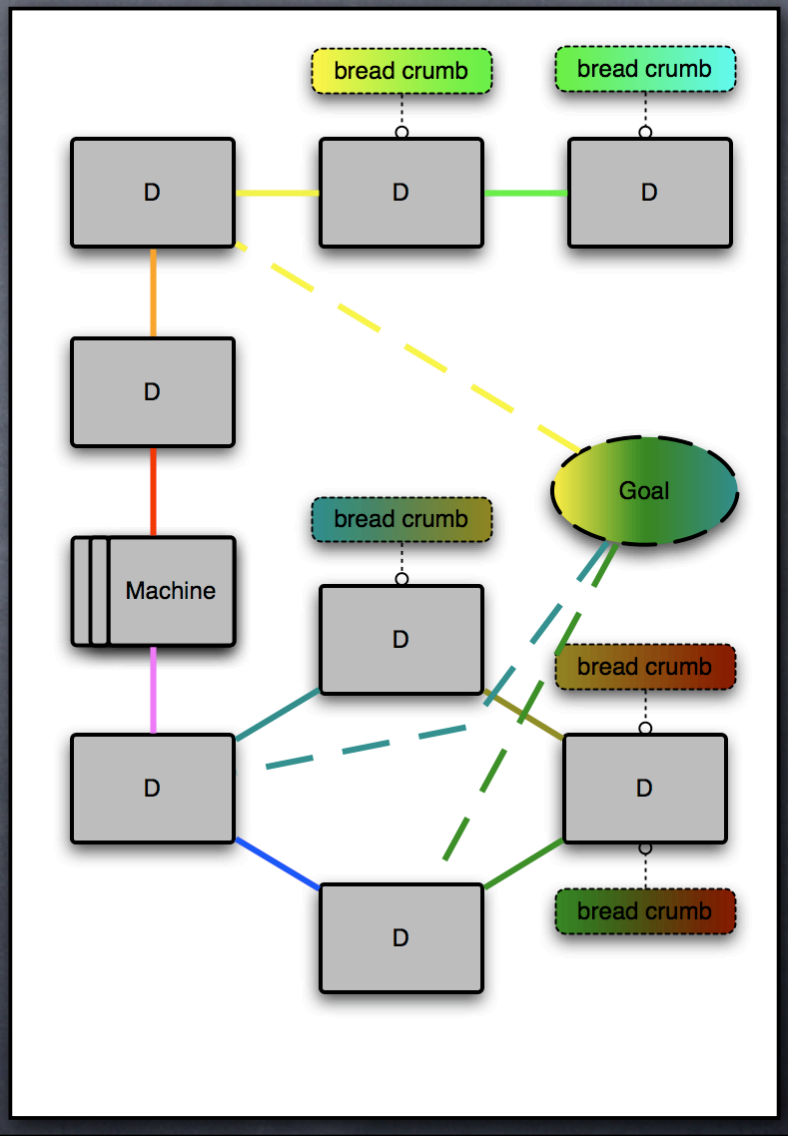
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



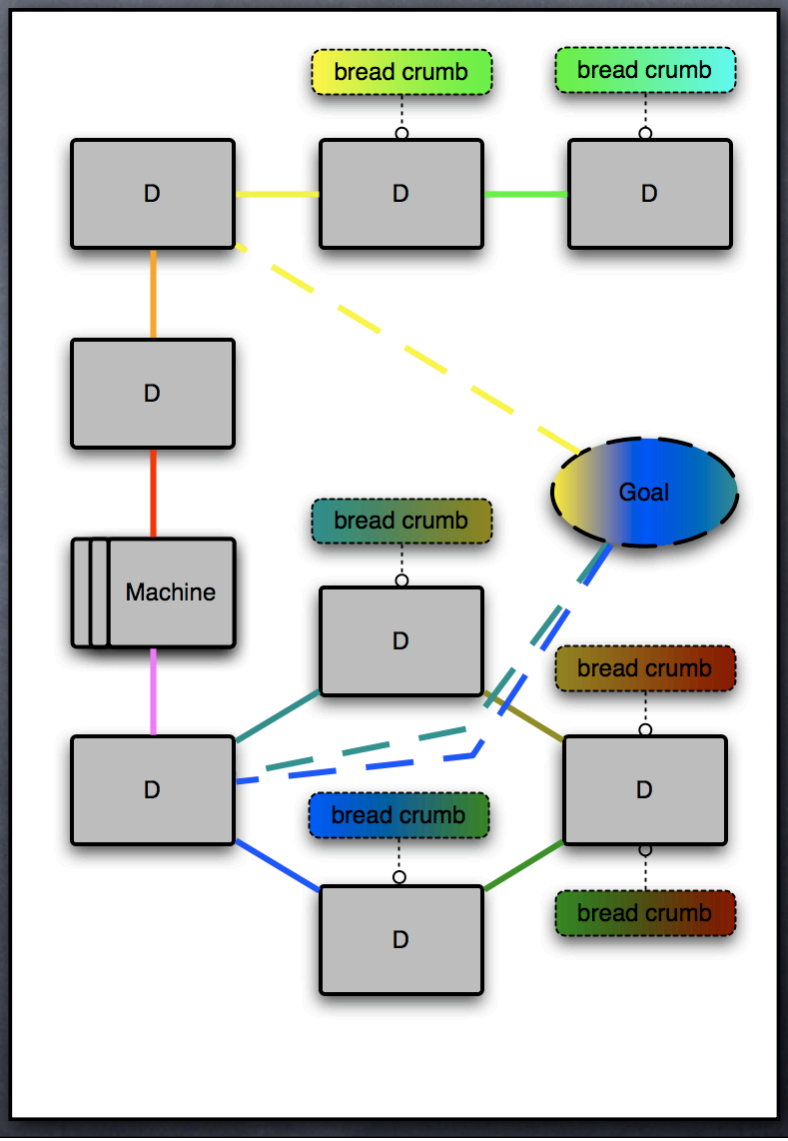
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



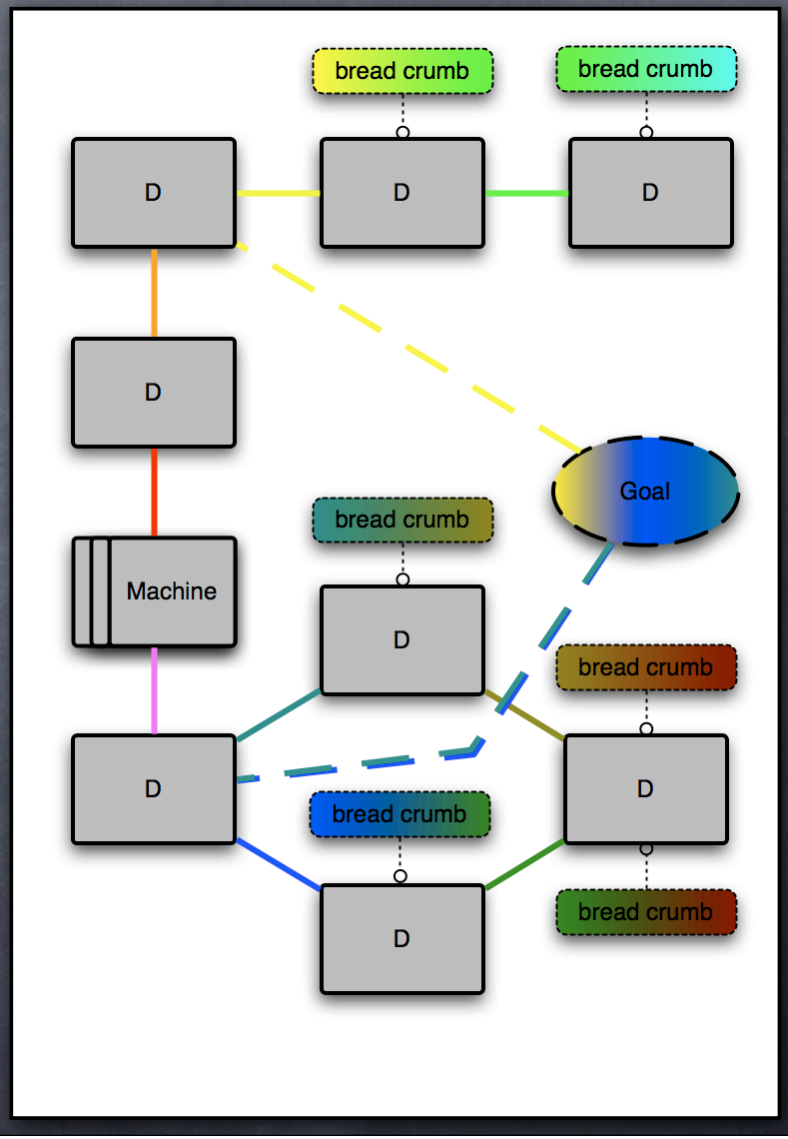
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



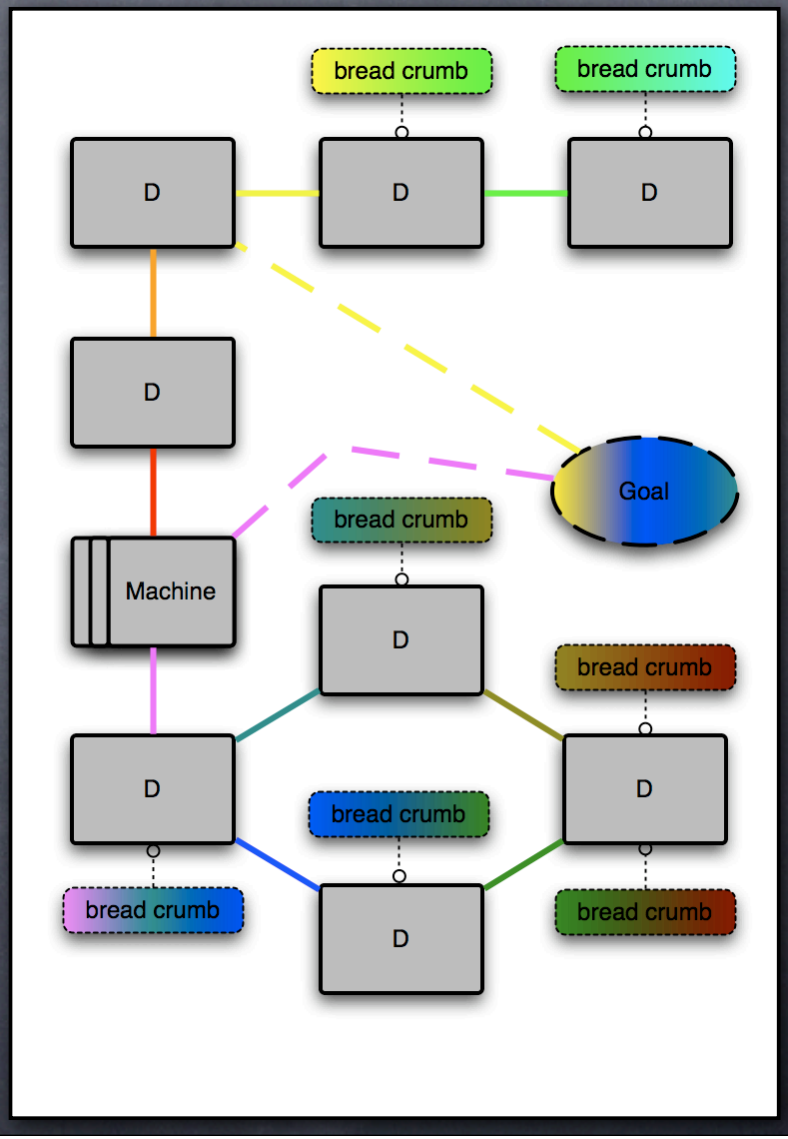
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



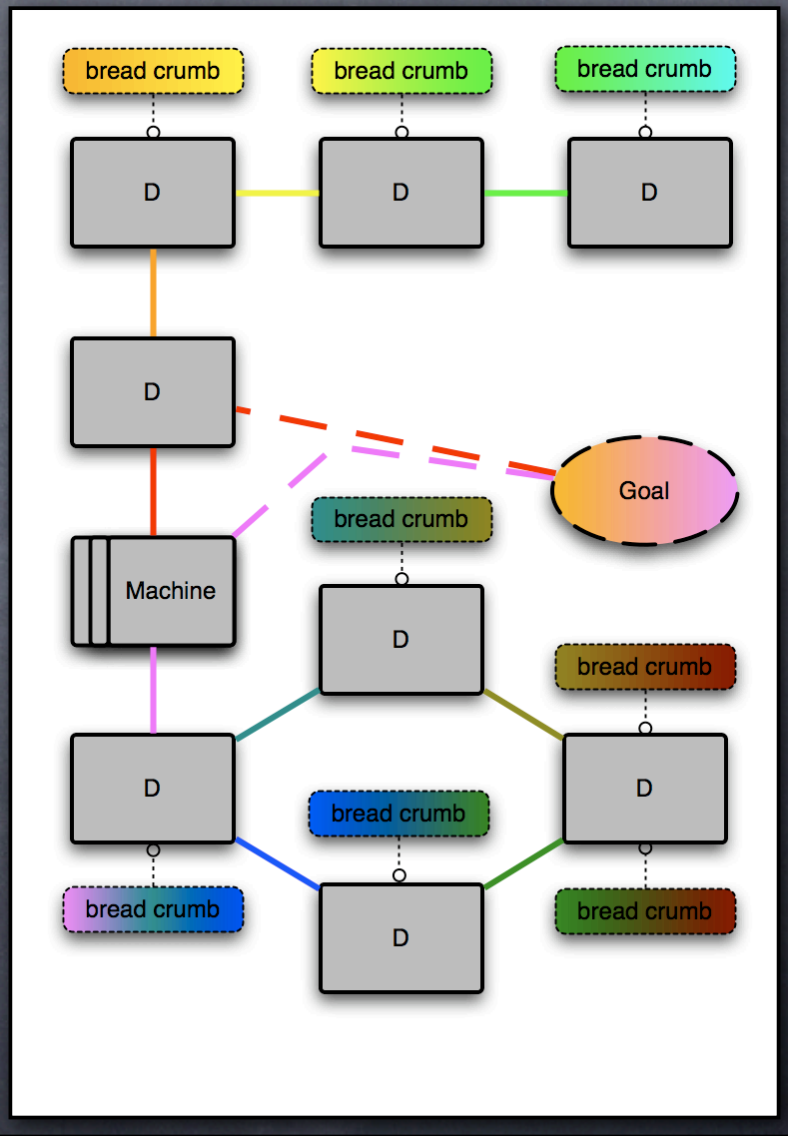
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



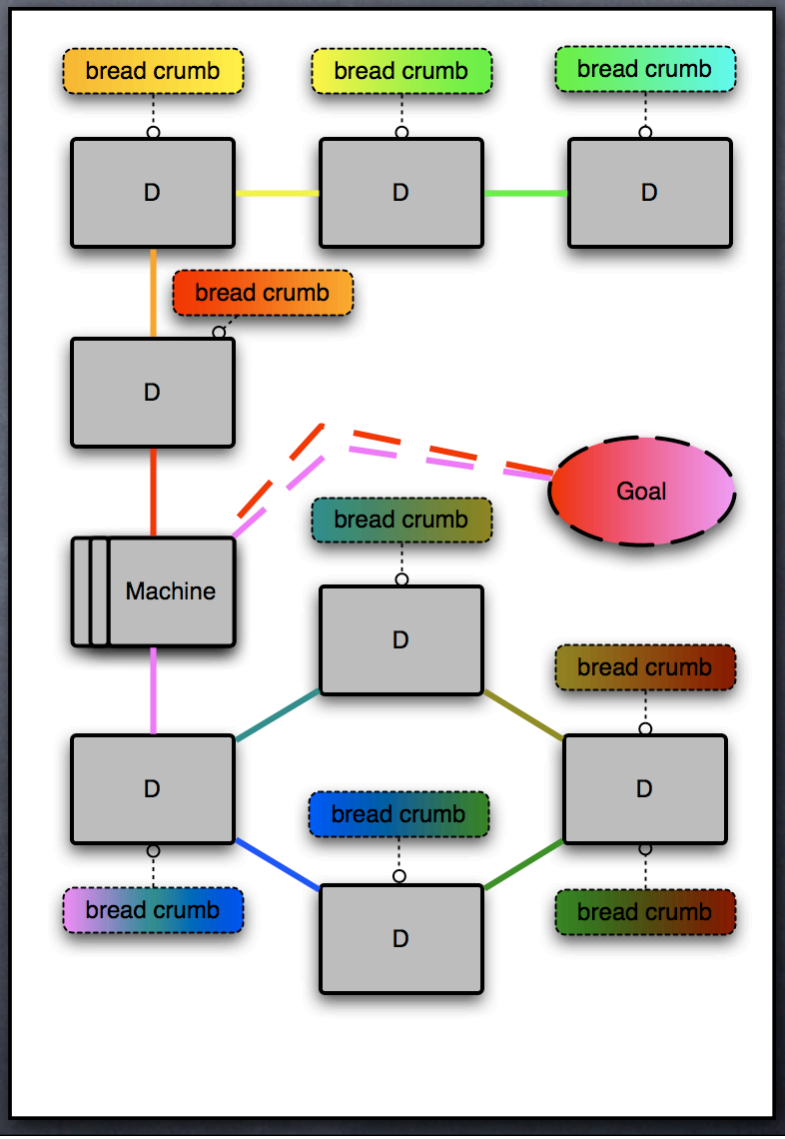
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



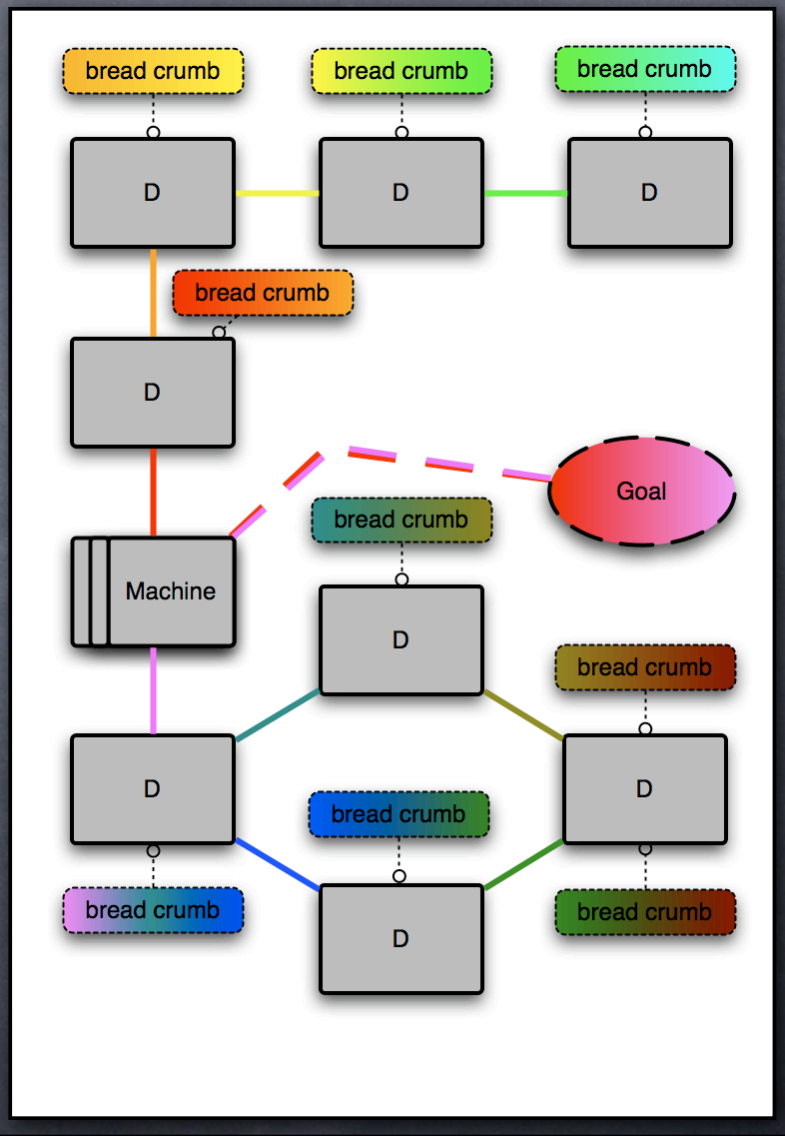
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



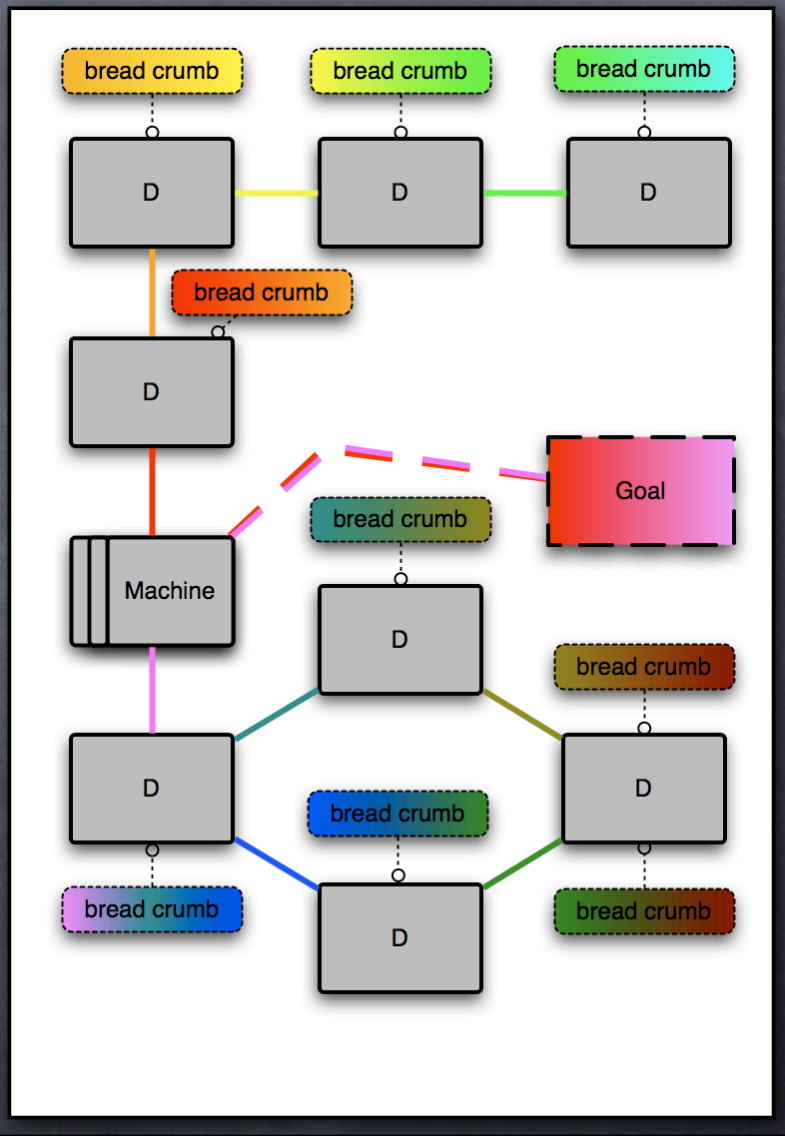
Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



Progression Toolkit

- Add Breadcrumb
  on phenomena of 1 domain
- Rephrase Requirement
  new ∧ breadcrumb ⇒ old
- Push Arc
  if phenomena are shared
- Split/Merge Arc
  nothing else changes
- Heuristics
  add / rephrase / push
  walk towards machine
  guided by informal "frame"



Traffic Light: Alloy Description

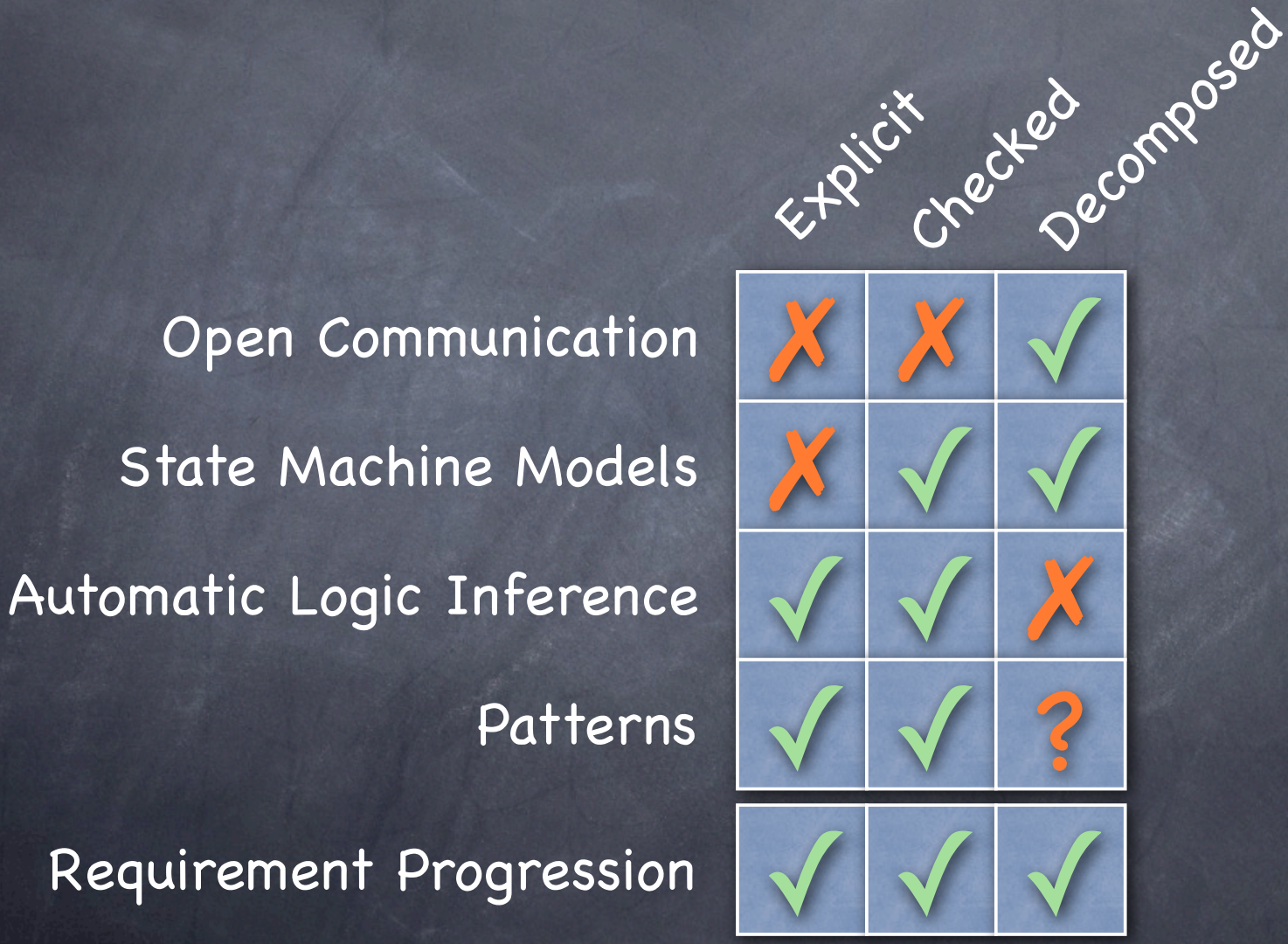
```
sig Cars {
   onSeg: time,
   dir: Direction -> time
}
pred CarOnSegment(c: Cars, t: time) { t in c.onSeg }
fun CarDirection(c: Cars, t: time) : Direction { (c.dir).t }

abstract sig Direction {}
one sig north, south extends Direction {}
sig time {}

sig NGObs, SGObs in time {}
pred NGO(t: time) {t in NGObs}
pred SGO(t: time) {t in SGObs}
```

Traffic Light: Alloy Check

```
pred old() {
   no t: time | some c1, c2 : Cars |
      CarDirection(c1, t) = north and
      CarDirection(c2, t) = south and
      CarOnSegment(c1, t) and
      CarOnSegment(c2, t)
}

pred new() {
   no t: time |
      NGO(t) and
      SGO(t)
}
```

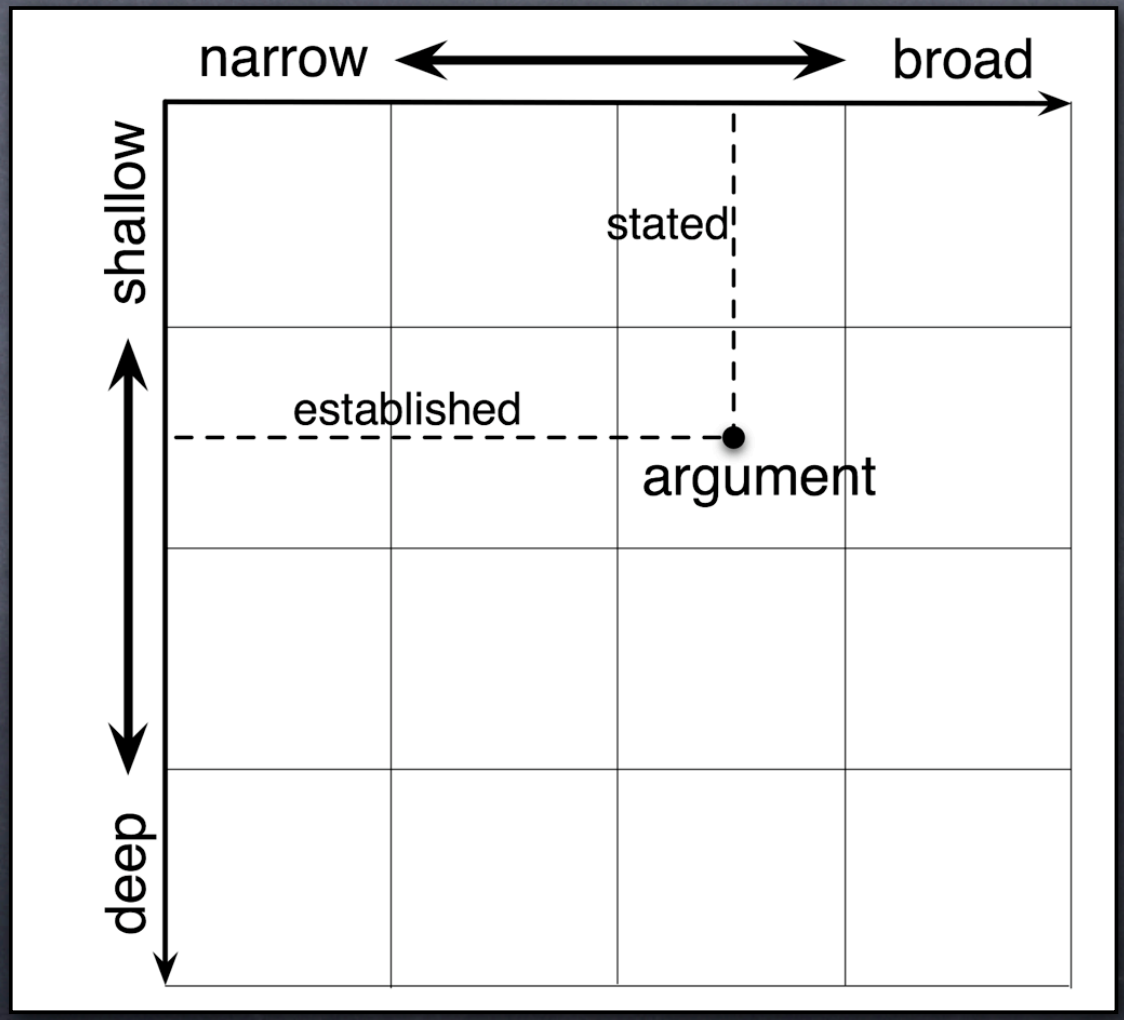
```
pred breadcrumb() {
   all t: time |
      ! NGO(t) =>
      (no c: Cars |
         CarDirection(c, t) = north and
         CarOnSegment(c,t))
   (all t: time |
      ! SGO(t) =>
      (no c: Cars |
         CarDirection(c, t) = south and
         CarOnSegment(c, t))
   )
}

assert valid {
   new() and breadcrumb() => old()
}
check valid for 10
```

Alternate Approaches

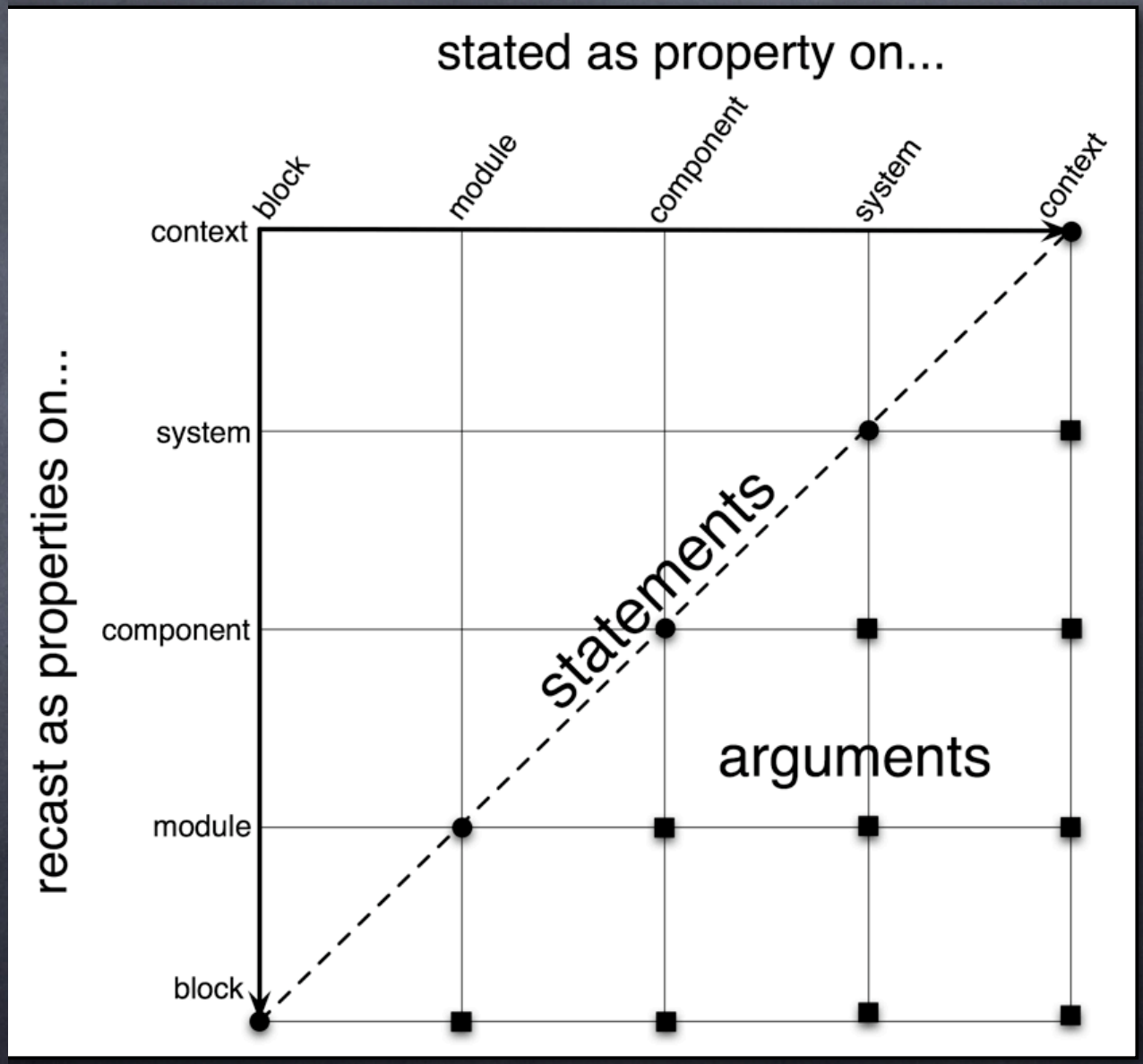


CDADs

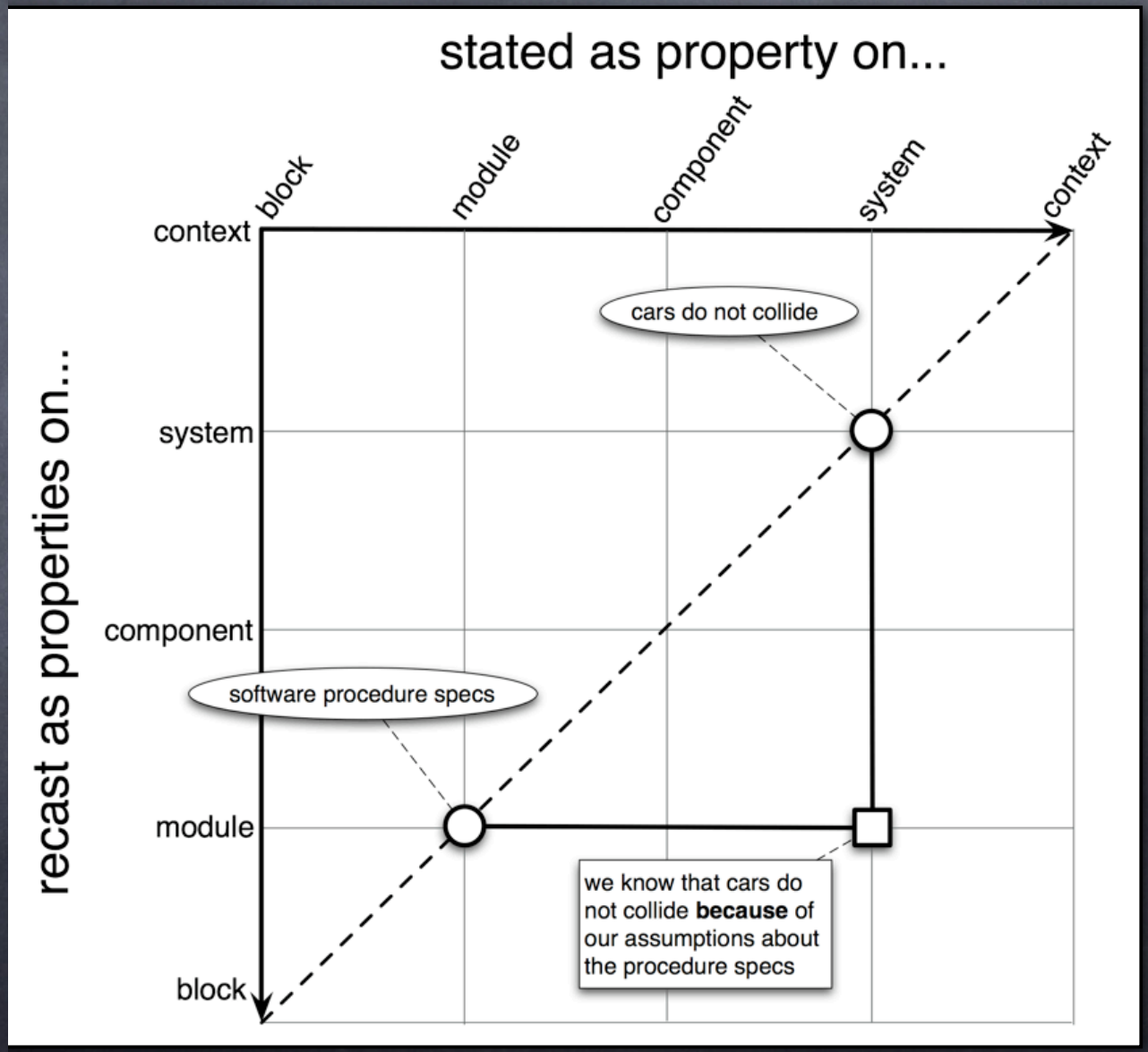
Composite Dependability Argument Diagrams



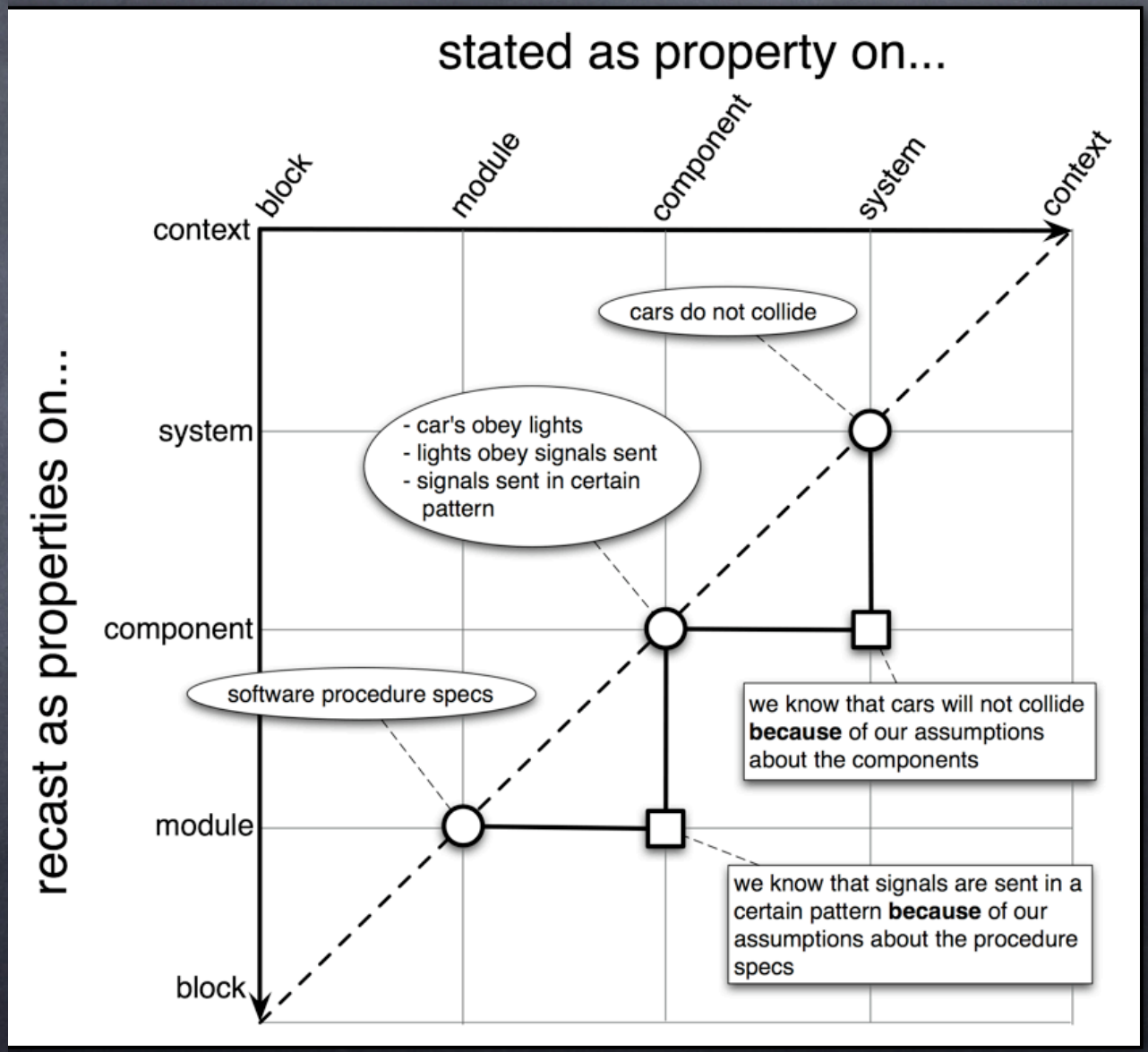
Interpretation



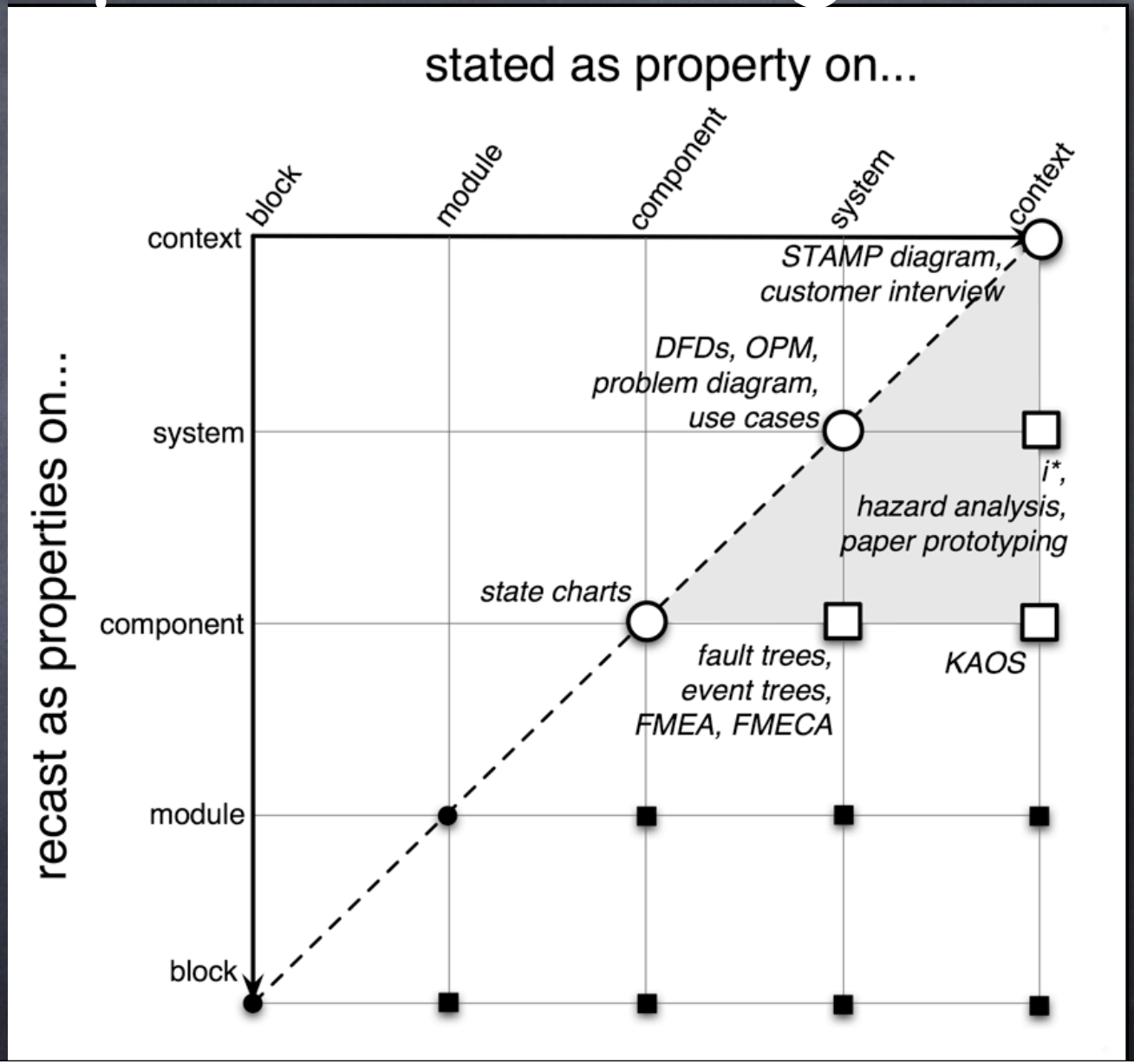
Traffic Light



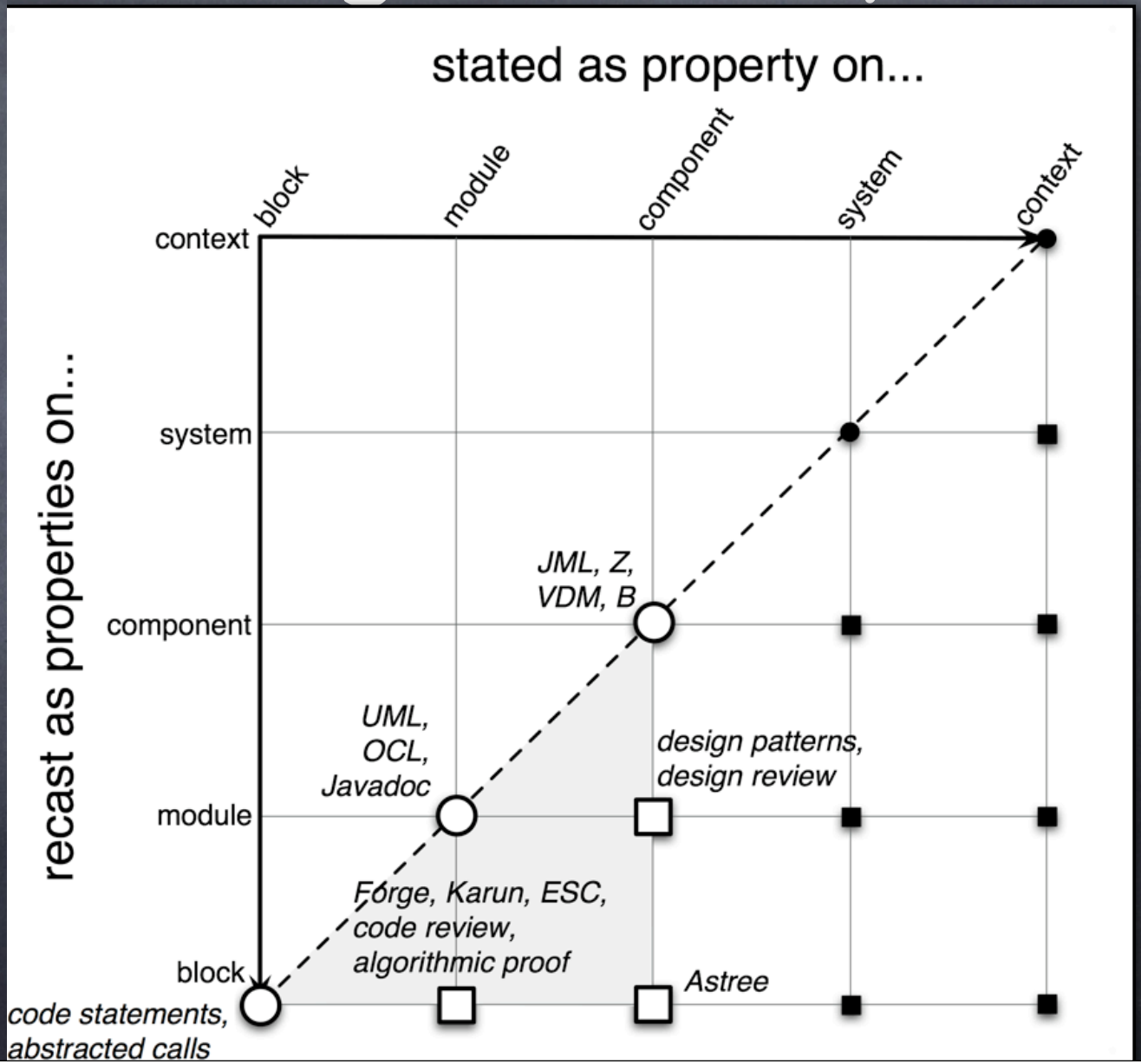
Traffic Light



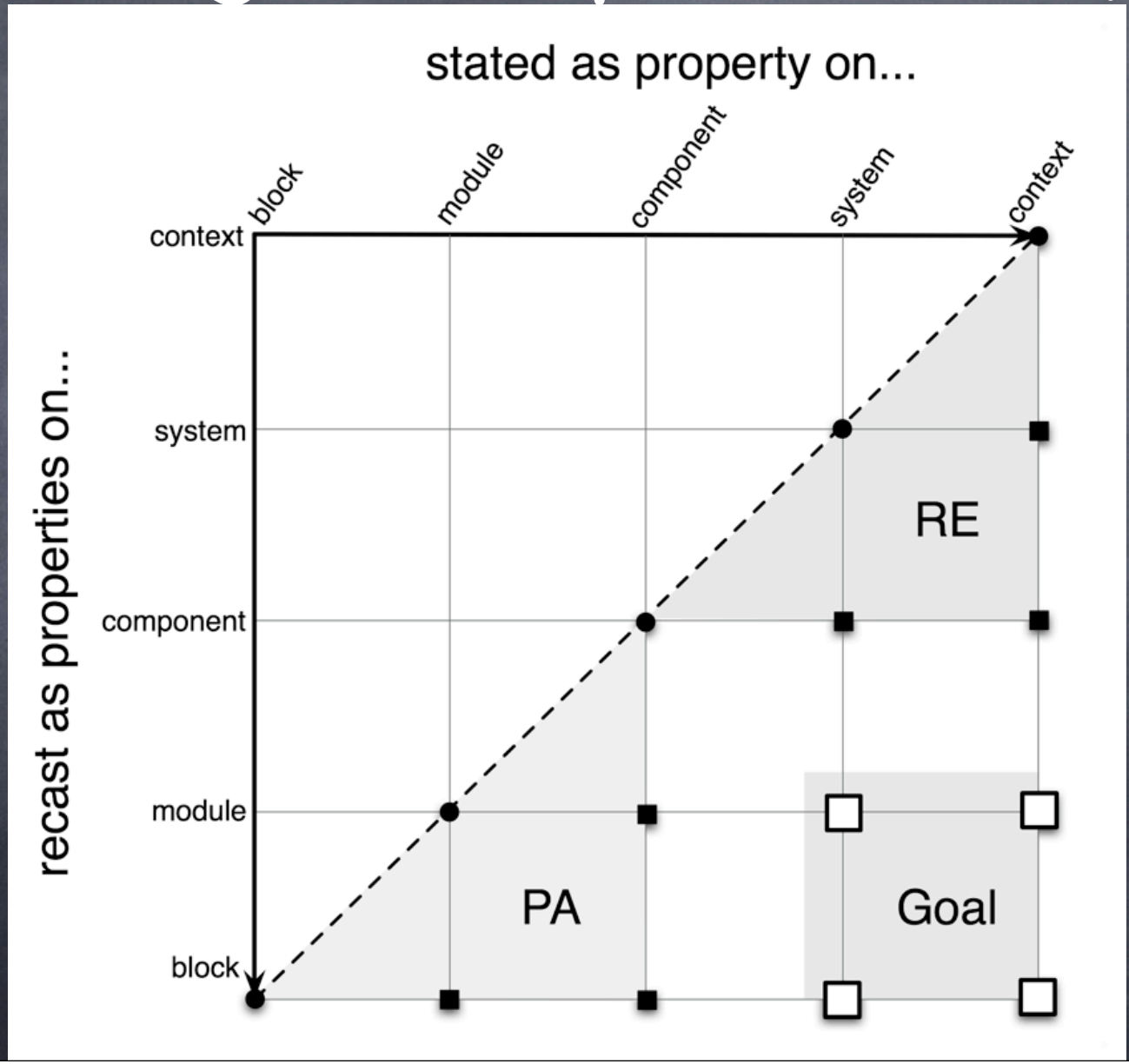
Requirements Engineering



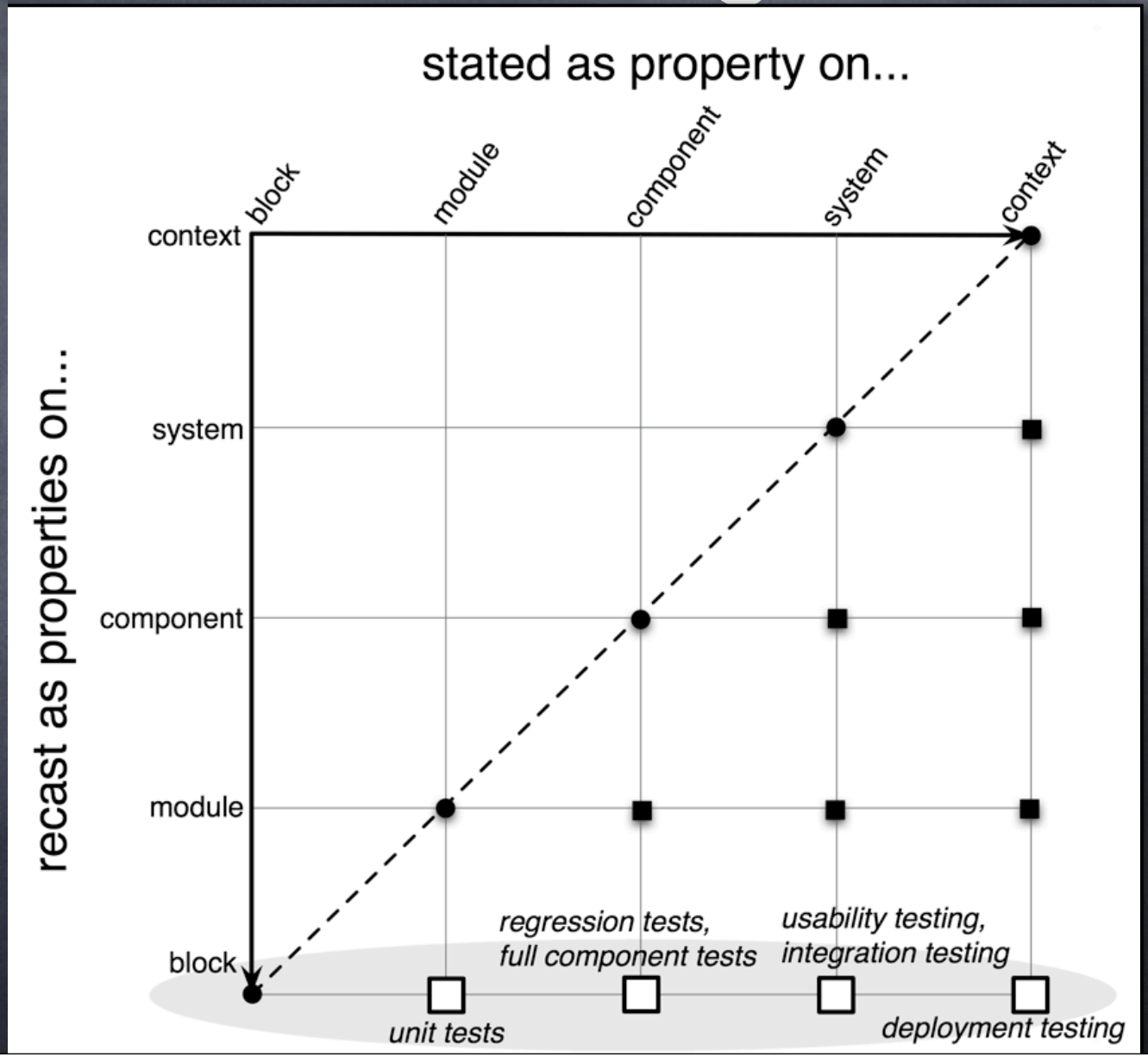
Program Analysis



Target: Dependability



Testing

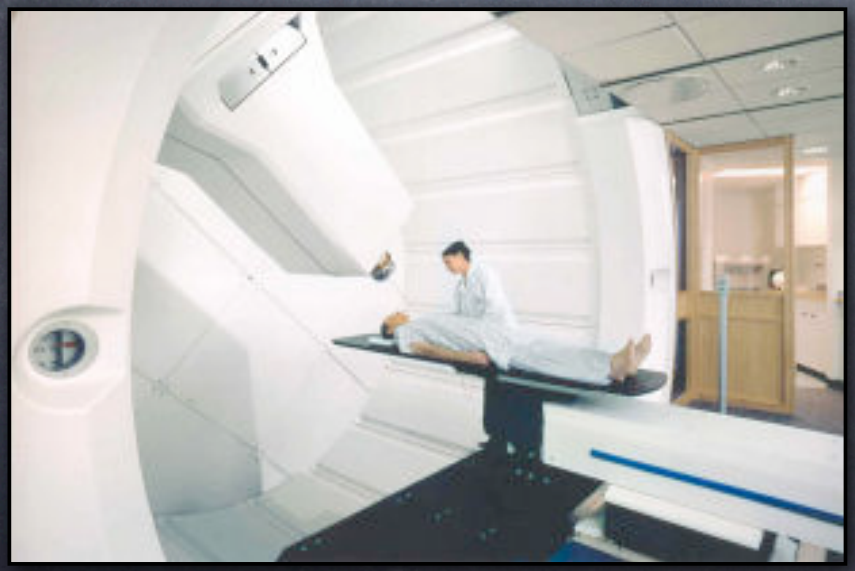

stated as property on...
block
module
component
system
context
context
system
component
module
block
recast as properties on...
regression tests,
full component tests
usability testing,
integration testing
unit tests
deployment testing

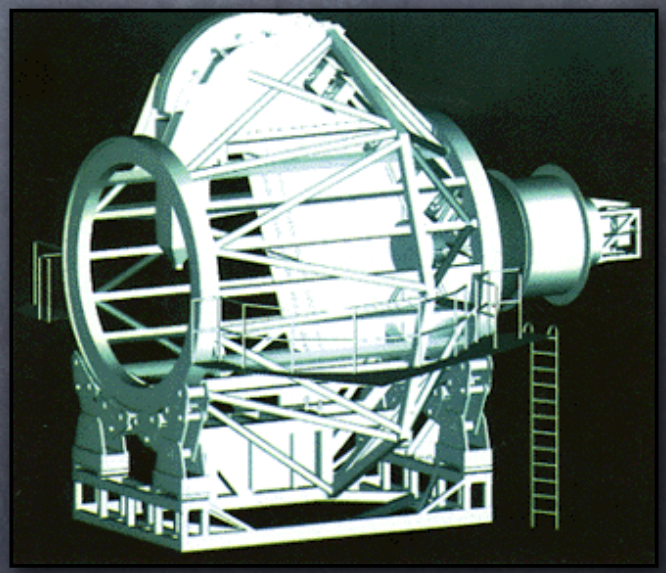
Composition

Burr Proton Therapy Center

MGH BPTC facility

- treating cancer patients since 2002
- high energy proton beam
- irradiate tumor, low collateral dmg
- precise enough for brain/eye/child





BPTC Requirements

delivery
- intensity
- location
- patient ←

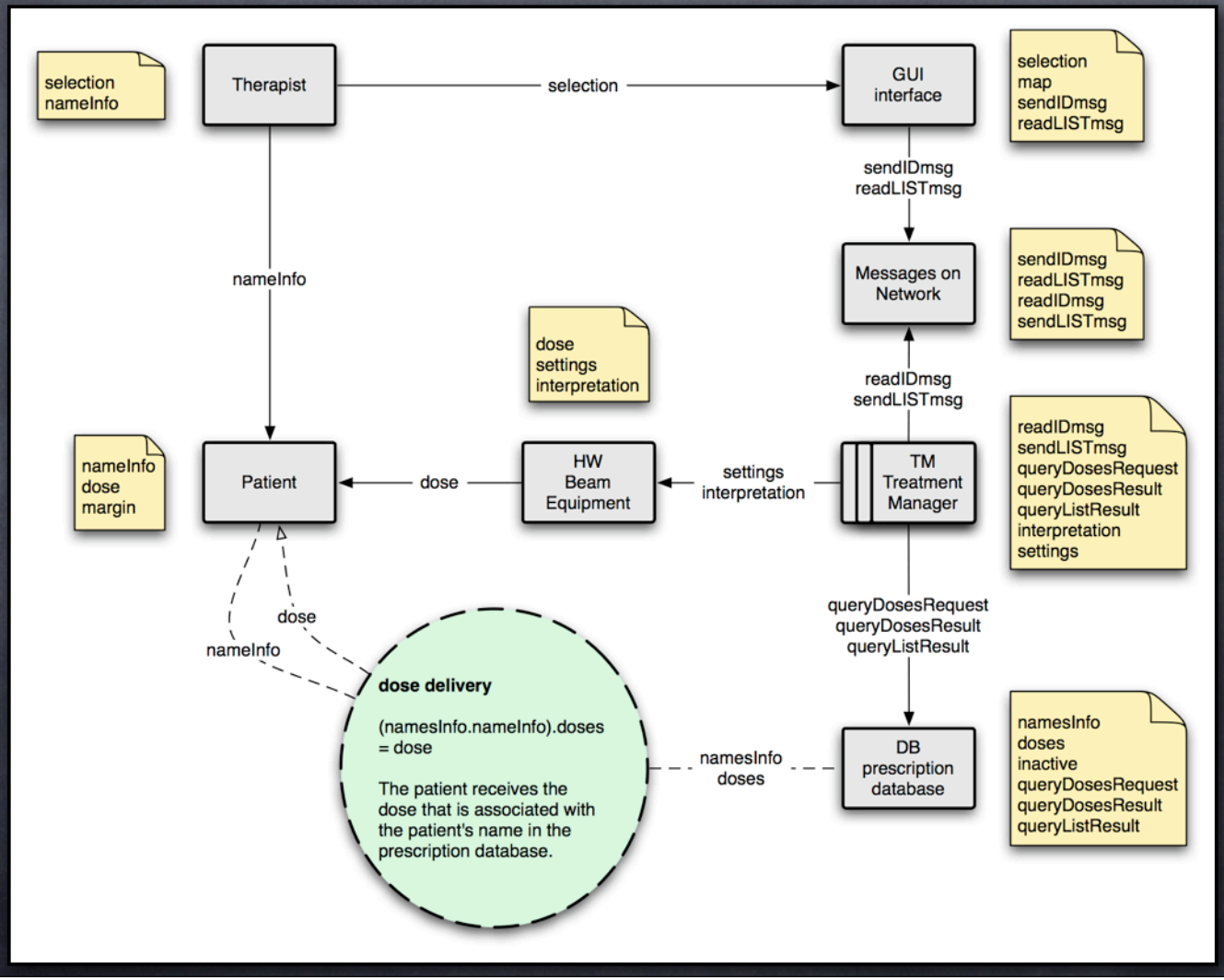
bookkeeping
- dose logging
- session logging
- privacy

operational
- throughput
- cost
- safe failure

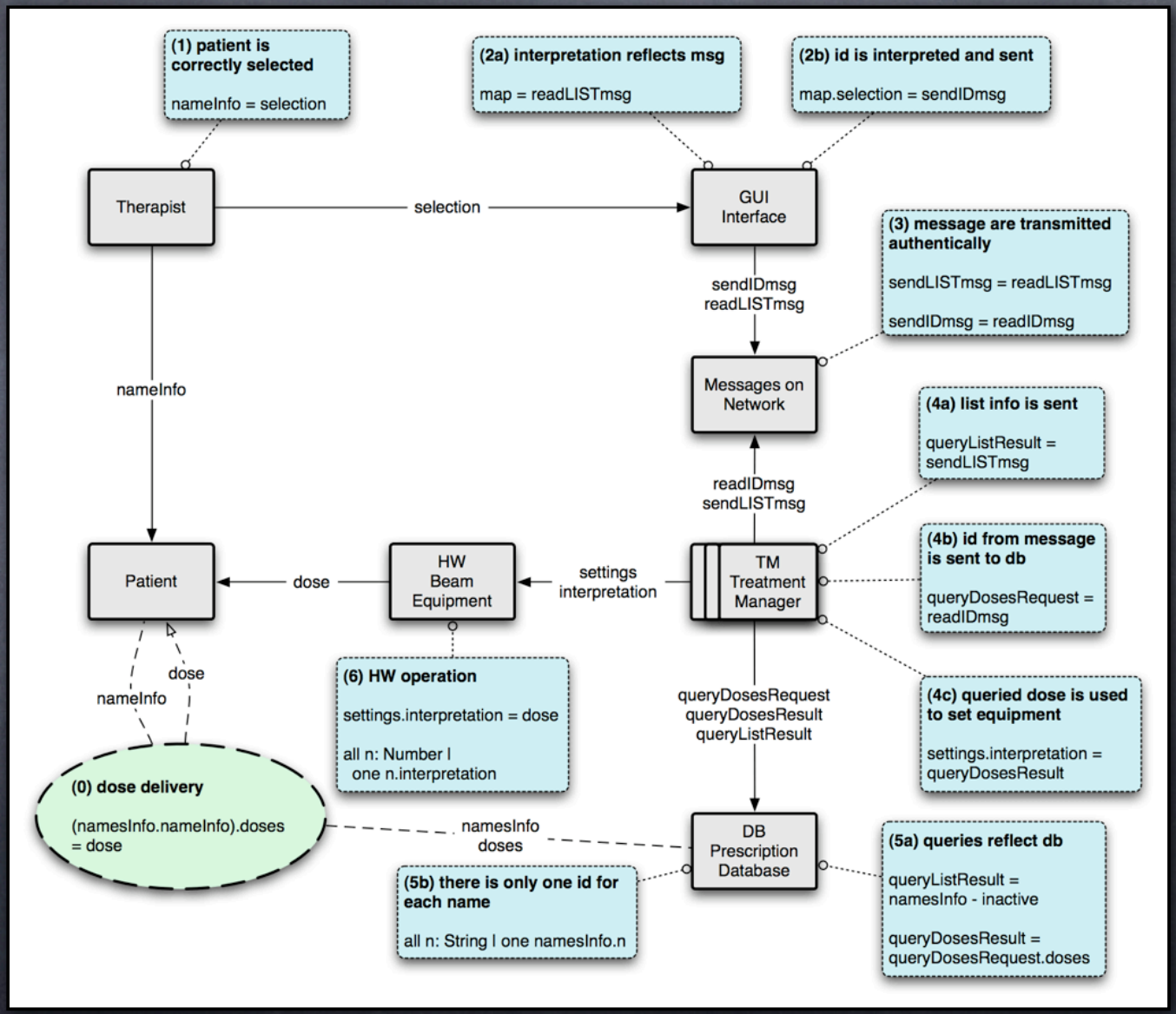
Our Focus: Dose Delivery

The patient received a dose of radiation with the intensity recorded in the Rx database under that patient's ID.

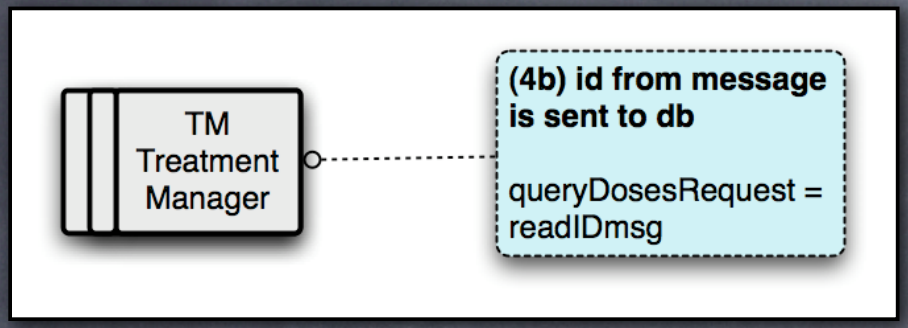
Problem Diagram



Argument Diagram



Code Analysis of TM



The ID from the message received from the GUI is the same ID used to query the DB for dose information.

correctness

- the ID is read from the message and stored in the same global used to generate the DB dose query

separability

- that global is not overwritten before the query is generated
- permits analysis of less code

Correctness

examine designation

A message is received as a pair of parameters, data and arg. The data parameter contains a message which is an array of identifiers. The 0th slot of that array indicates the screen that was displayed when the message was generated. The 1th slot indicates the button that was pressed to trigger the message. The arg parameter is a lump of data containing state information about the gui. Part of that lump of data is the identity of the patient being treated.

interpret into code terminology

```
data.data__msg.mixed_array_index[0] == SCR_A1_PATIENT_SELECTION
&& data.data__msg.mixed_array_index[1] == W_PATIENT_SELECT_BTN
=> current_id_patient == arg.scrCrtPatientData.dbs_patient_type__id_patient
```

pass to Forge (Java API)

```
final ForgeExpression correct_result =
    current_id_patient.eq(arg.join(scrCrtPatientData)
                          .join(dba_patient_type__id_patient))
    .and(
        one.join(data.join(data__msg).join(mixed_array_index))
        .eq(W_PATIENT_SELECT_BTN))
    .and(
        zero.join(data.join(data__msg).join(mixed_array_index))
        .eq(SCR_A1_PATIENT_SELECTION));
```

Forge

automatic analysis

- relational claims (Alloy)
- counter-example traces
- iterative precondition discovery

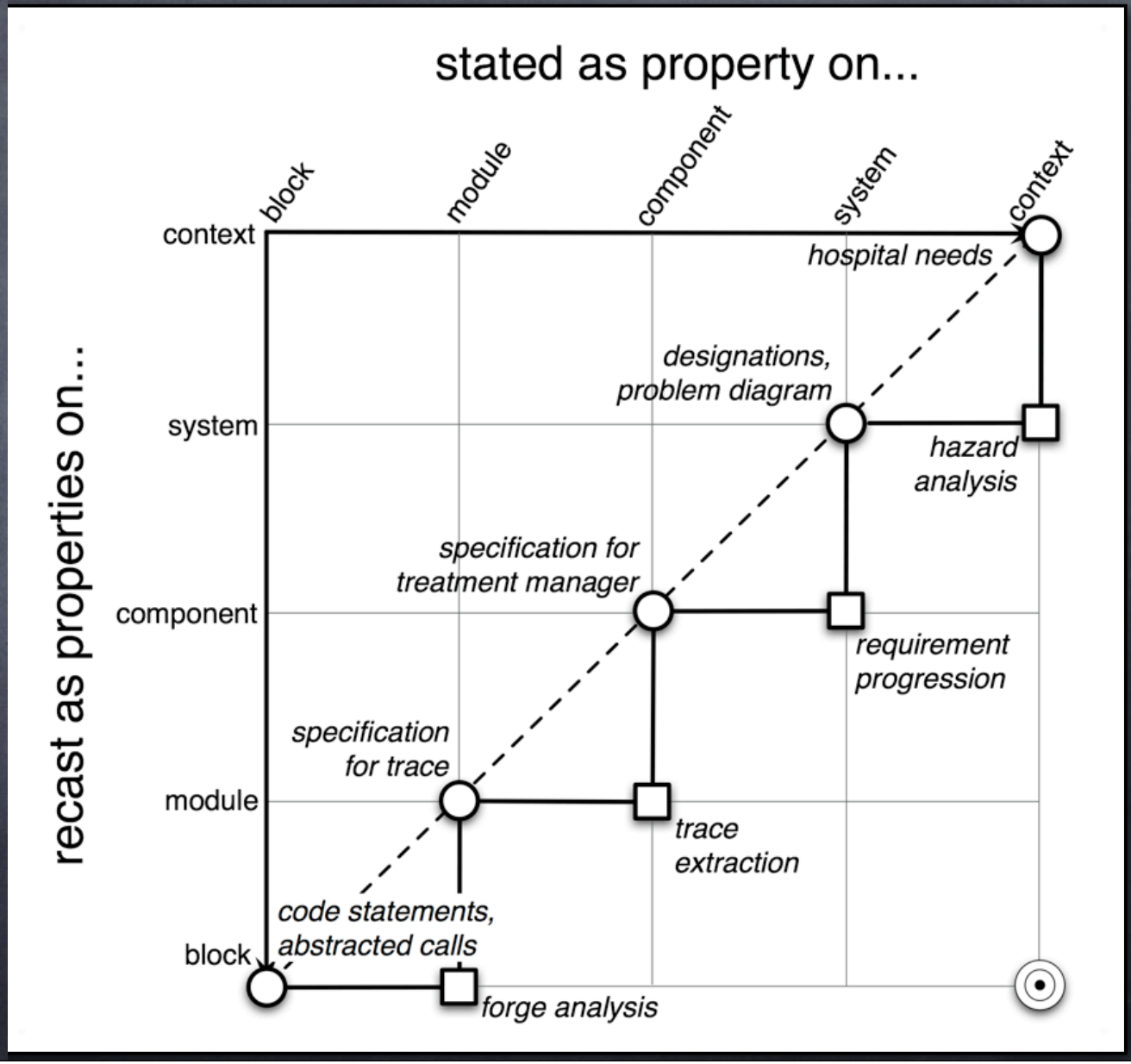
preprocessing: Forge Intermed. Rep.

- JForge, CForge
- manual translation to FIR

scalability: needs to add specs for

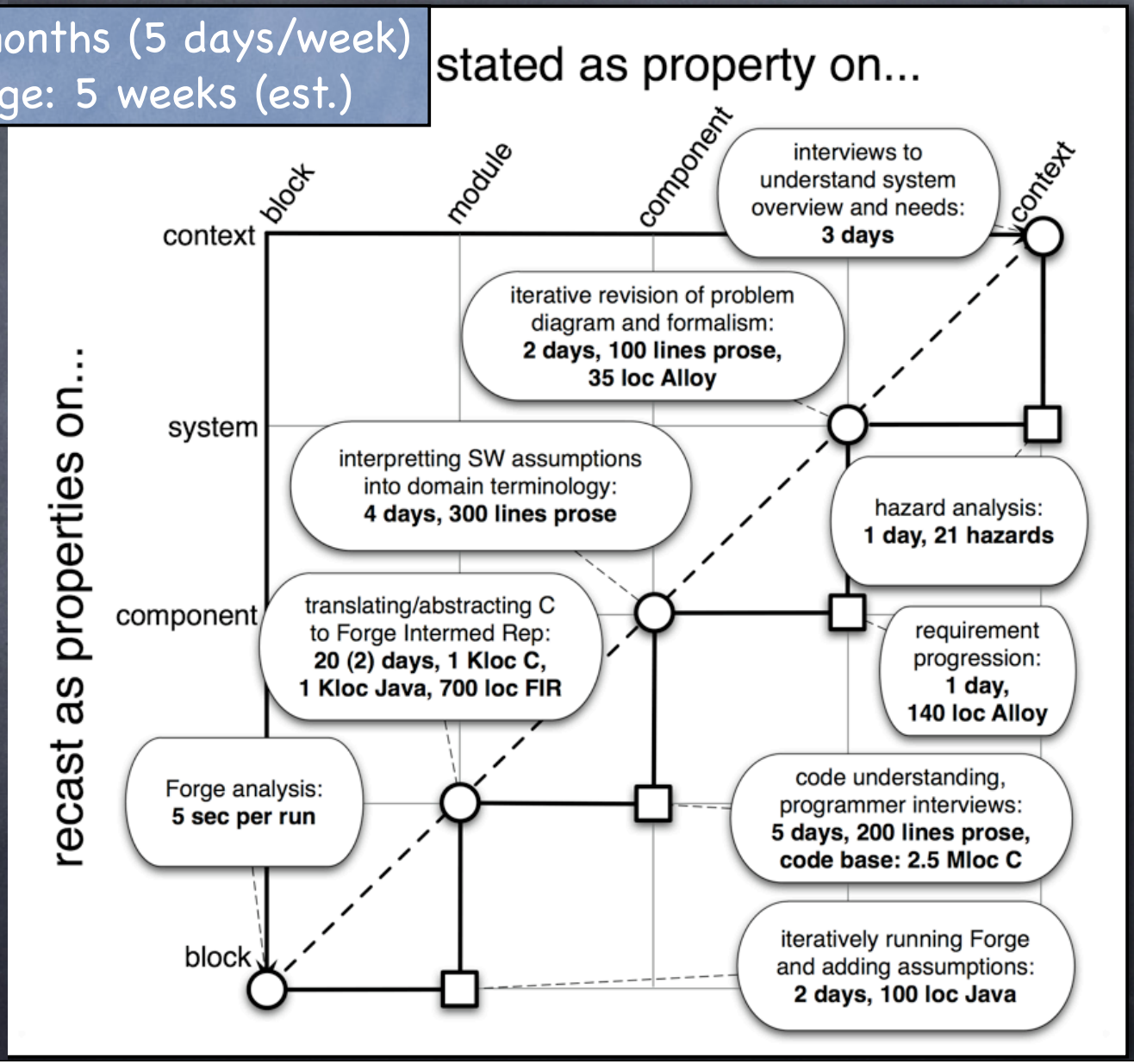
- library calls
- proprietary code
- cutoff points (reduce volume)
- reduce 2.5 Mloc -> 1 Kloc

CDAD for TM


stated as property on...
block
module
component
system
context
recast as properties on...
context
system
component
module
block
hospital needs
designations,
problem diagram
hazard
analysis
specification for
treatment manager
requirement
progression
specification
for trace
trace
extraction
code statements,
abstracted calls
forge analysis

Timetable

total: 2 months (5 days/week)
with CForge: 5 weeks (est.)



Achievements

safety argument for critical property

- traceable end-to-end argument
- found & documented assumptions
- reasonable cost - est. 1 year more

current vulnerabilities

- sql injection (separability)
- message delays
- patient identification process

future vulnerabilities

- network (dropped, reordered msg)
- database (format, units)
- GUI generation (authentic i/o)
- code (structure, redundant, globals)
- new firing mode

Limitations

vulnerabilities not errors

- sometimes find errors in process
- limited by quality of assumption confirmation
- hard to assess confidence of human domains

requires skilled analysis & support

- support from domain specialists
- physicist, doctor, therapist, programmer, manager, operator (not patients)
- trouble with staff changes
- relational logic is easy...to me

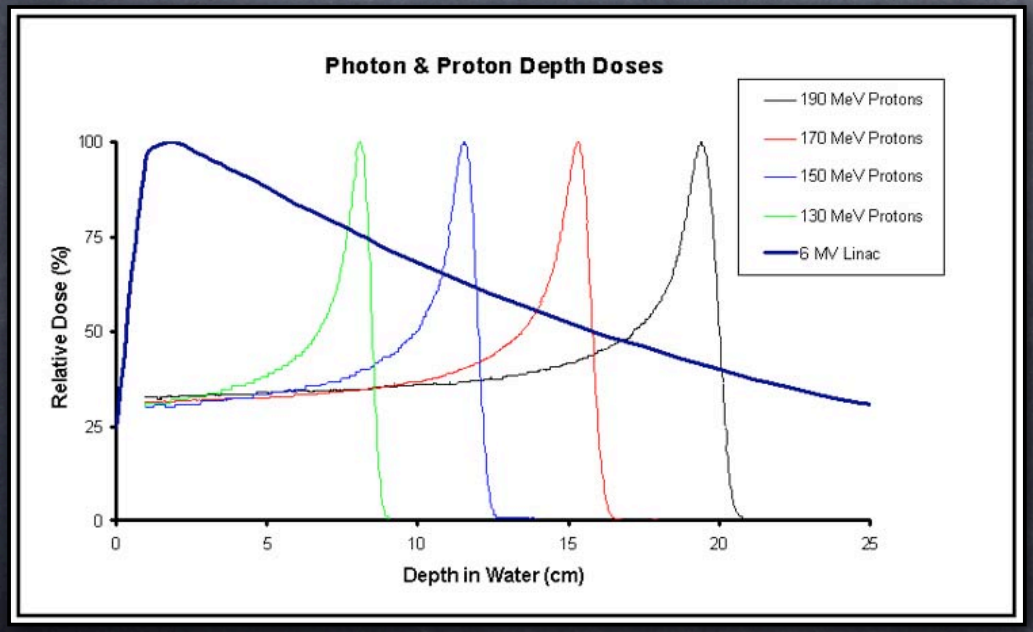
code analysis requires abstraction to scale

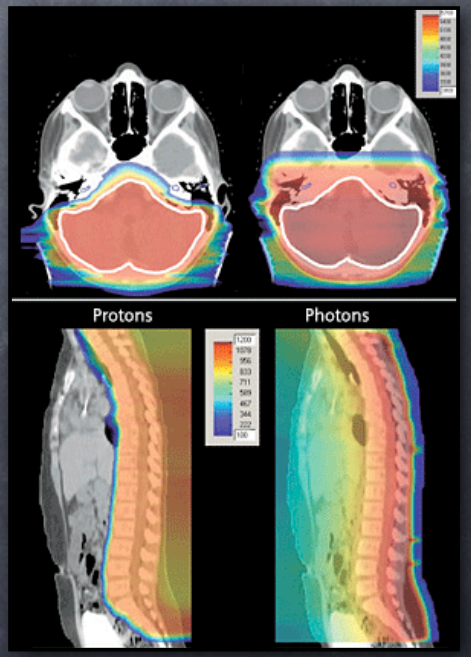
- poorly structured code
- scaling limitations of analysis

Burr Proton Therapy Center

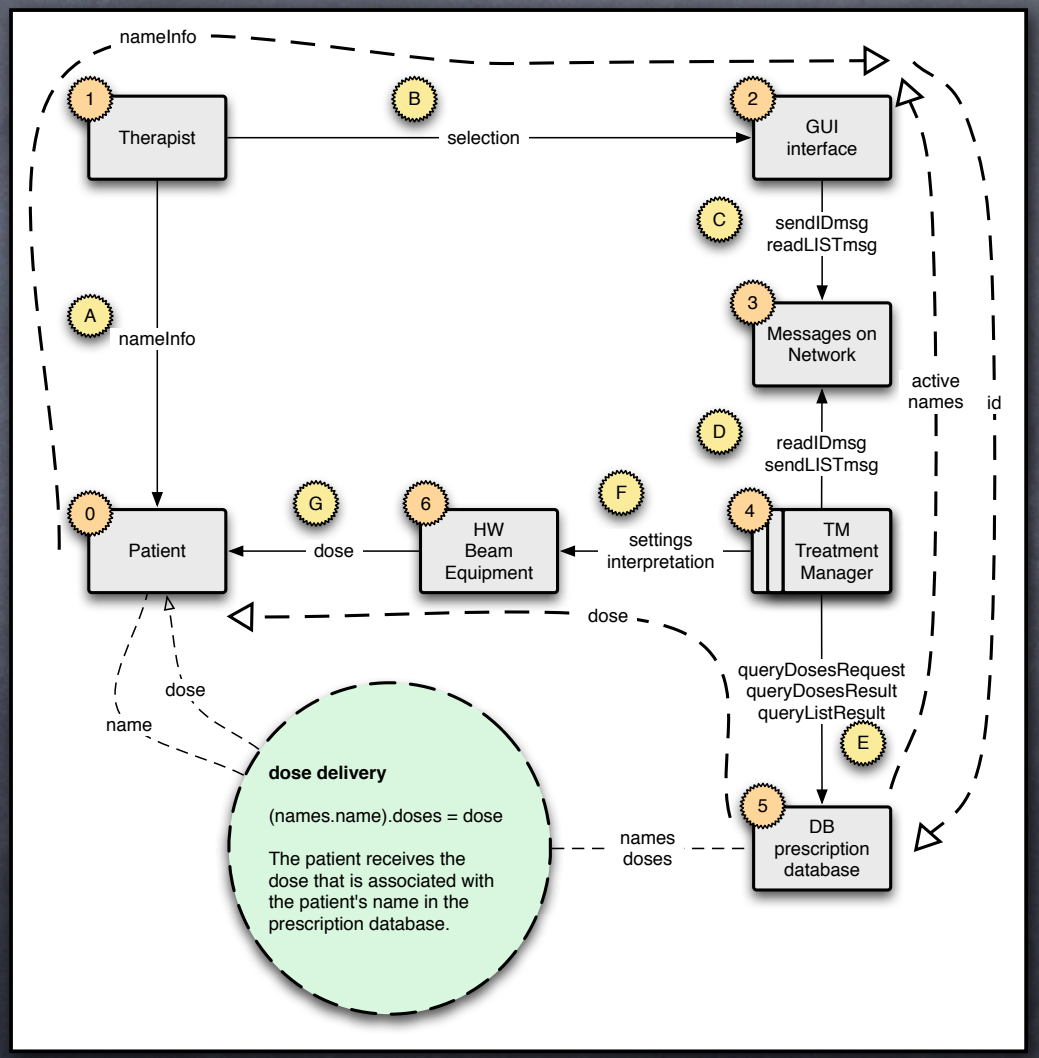
MGH BPTC facility

- treating cancer patients since 2002
- high energy proton beam
- irradiate tumor, low collateral dmg
- precise enough for brain/eye/child





Flow Diagram



Identification Procedure

sharing assumptions: NameInfo

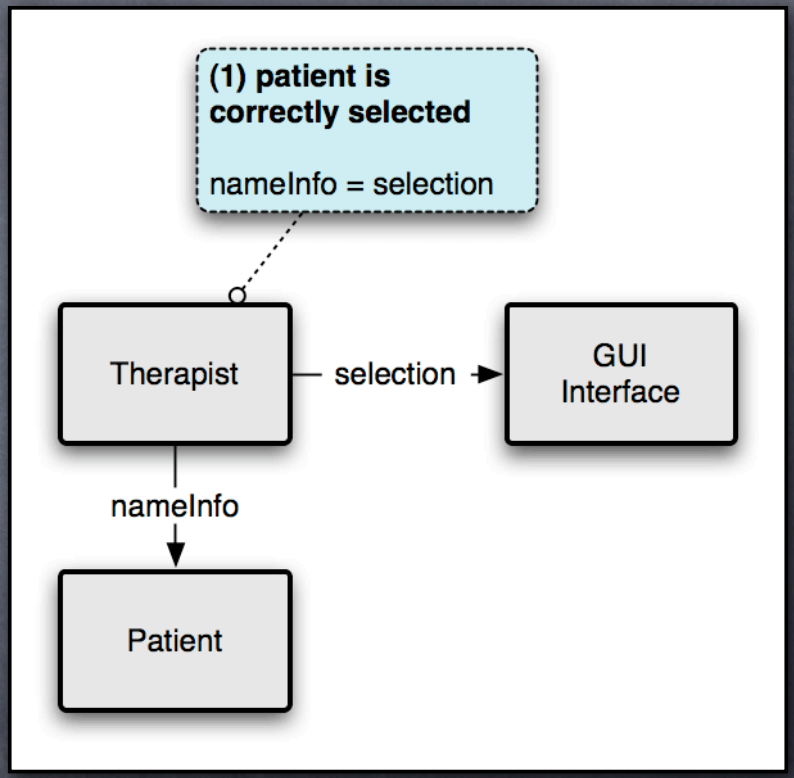
- unique name
- correctly read from tag

sharing assumptions: selection

- unique
- confirmed (mis-click)
- GUI SW error

domain assumptions

- memory while back turned
- alternate spelling
- duplicate entries on scroll list



CODE TRANSLATION

Original C

```
/*****************************************************************************
* tpcrInSelectPatient
* PSEUDO-CODE :
*
* Extracts groups of information in array.
* Checks value of property field.
* Checks value of widgetGroupId field.
* Checks value of widgetId field.
* Extracts patient id and copies to structure scrCrtPatientData.
* Calls function eventsTPCRSelectPatient.
*
*/
BOOLEAN tpcrInSelectPatient(
            /* IN */       T_INT4 screenId,
            /* IN */       T_INT4 sizeofList,
            /* IN */       DATA_MSG_GROUP_ARRAY msgList,
            /* IN_OUT */   DBASCR_SCR_DATA_PTR_TYPE pscrData)
{
   T_INT4 num;

   /* Process list of message */
   for(num = 0; num < sizeofList; num++) {
      if(msgList[num].property != PDEF_CONTENTS_PROPERTY) {
         SW_ERROR_MSG(ERR_APP_WRONG_PROPERTY_TYPE);
         return FALSE;
      }

      switch(msgList[num].widgetGroupId) {

         case WG_PATIENT:
            switch(msgList[num].widgetId) {
               case W_PATIENT_ID:
                  if(strlen(msgList[num].value) > DBA_PATIENT_ID_LEN) {
                     SW_ERROR_MSG(ERR_APP_WRONG_STR_VALUE);
                     return FALSE;
                  }
                  strcpy(pscrData -> scrCrtPatientData.id_patient, msgList[num].value);
                  break;
               default: /* Error happens */
                  SW_ERROR_MSG(ERR_APP_WRONG_WIDGET_ID);
                  return FALSE;
            } /* widgetId */
            break;

         default: /* Error happens */
            SW_ERROR_MSG(ERR_APP_WRONG_WIDGET_GROUP_ID);
            return FALSE;
      } /* widgetGroupId */
   }

   /* call events function correspond to "Select Patient" button */
   if(!eventsTPCRSelectPatient(screenId, pscrData)) { /* Error happens */
      TRACE_ERROR_MSG();
      return FALSE;
   }

   return TRUE;
}
```

Condensed C

```
BOOLEAN tpcrInSelectPatient(T_INT4 screenId, T_INT4 sizeofList,
        DATA_MSG_GROUP_ARRAY msgList, DBASCR_SCR_DATA_PTR_TYPE pscrData) {
   T_INT4 num;
   for(num = 0; num < sizeofList; num++) {
      if(msgList[num].property != PDEF_CONTENTS_PROPERTY) {
         SW_ERROR_MSG(ERR_APP_WRONG_PROPERTY_TYPE);
         return FALSE; }
      switch(msgList[num].widgetGroupId) {
         case WG_PATIENT:
            switch(msgList[num].widgetId) {
               case W_PATIENT_ID:
                  if(strlen(msgList[num].value) > DBA_PATIENT_ID_LEN) {
                     SW_ERROR_MSG(ERR_APP_WRONG_STR_VALUE);
                     return FALSE; }
                  strcpy(pscrData -> scrCrtPatientData.id_patient, msgList[num].value);
                  break;
               default:
                  SW_ERROR_MSG(ERR_APP_WRONG_WIDGET_ID);
                  return FALSE; }
            break;
         default:
            SW_ERROR_MSG(ERR_APP_WRONG_WIDGET_GROUP_ID);
            return FALSE; } }
   if(!eventsTPCRSelectPatient(screenId, pscrData)) {
      TRACE_ERROR_MSG();
      return FALSE; }
   return TRUE;
}
```

Abstracted C

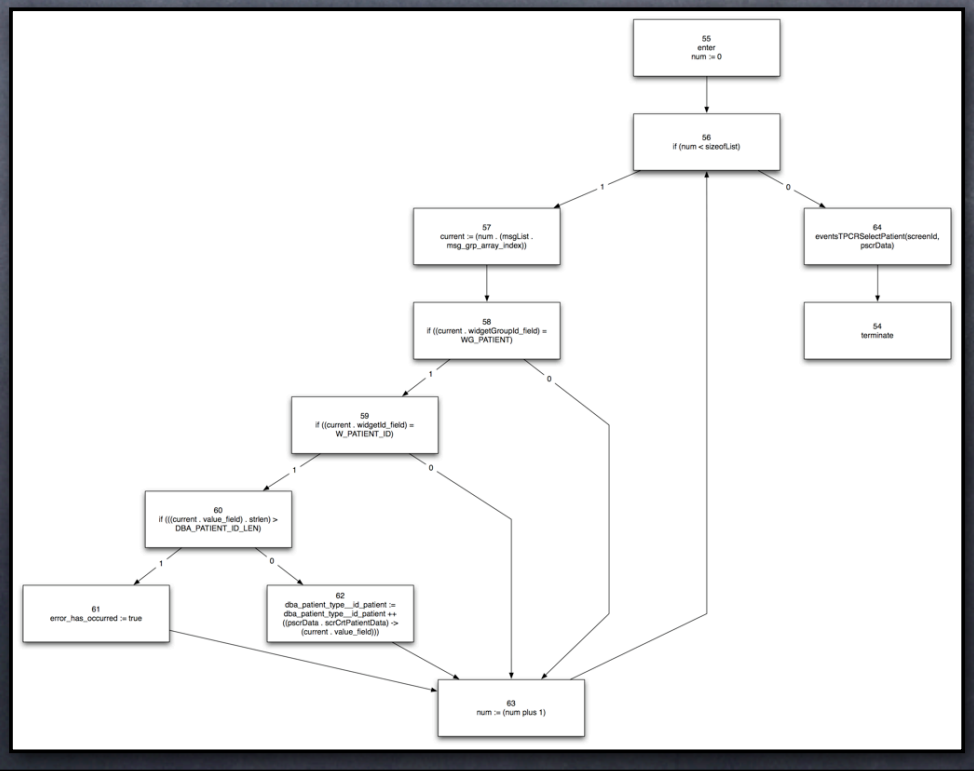
```
void tpcrInSelectPatient (
    T_INT4 screenId,
    T_INT4 sizeofList,
    DATA_MSG_GROUP_ARRAY msgList,
    DBASCR_SCR_DATA_PTR_TYPE pscrData)
{
    T_INT4 num;
    for(num = 0; num < sizeofList; num++) {
      if (msgList[num].widgetGroupId =  WG_PATIENT)
        if (msgList[num].widgetId = W_PATIENT_ID)
          if (strlen(msgList[num].value) > DBA_PATIENT_ID_LEN)
            ERROR;
          else
            strcpy(pscrData -> scrCrtPatientData.id_patient, msgList[num].value);
    }
    eventsTPCRSelectPatient(screenId, pscrData);
}
```

Java

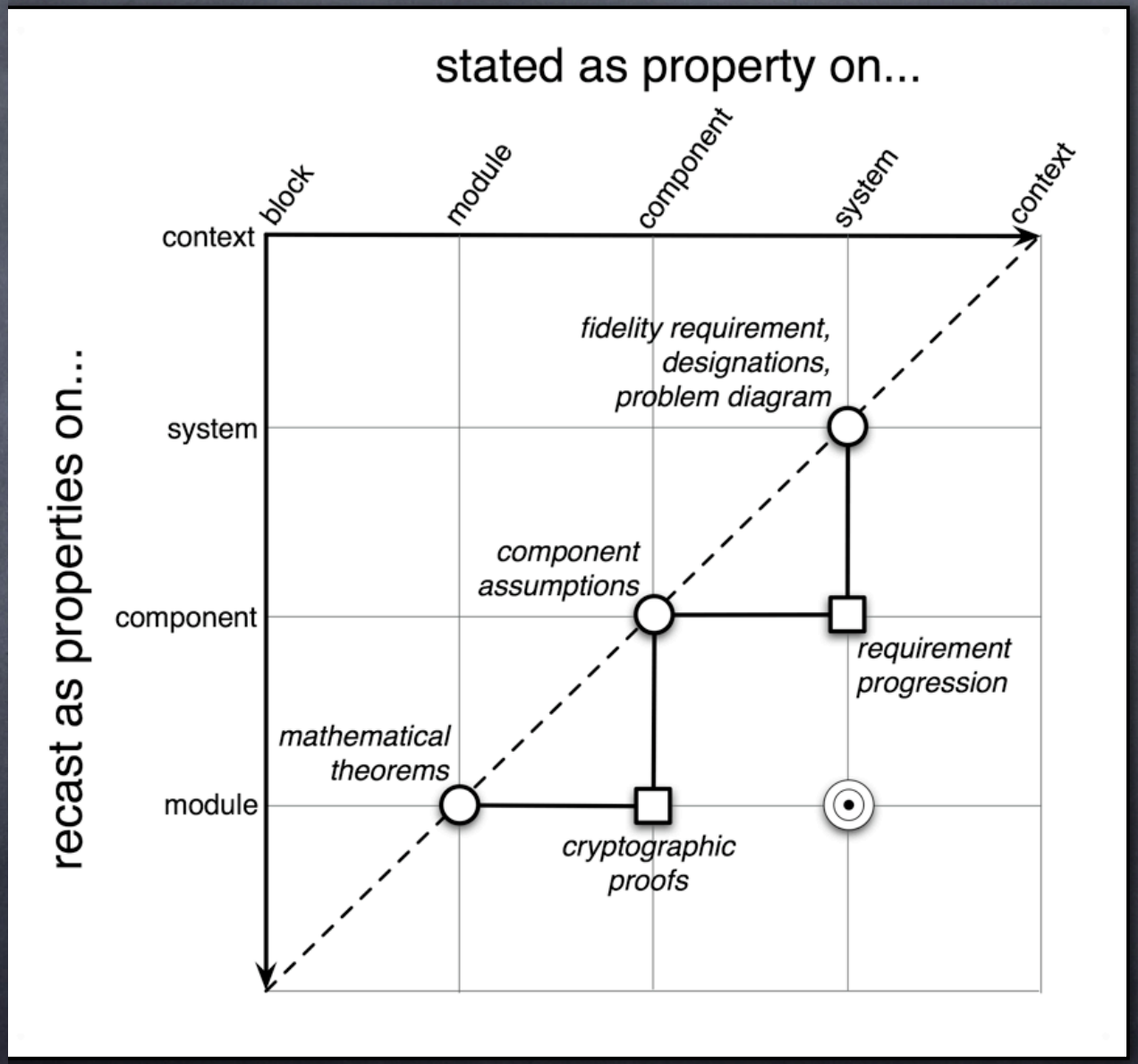
```
void define__tpcrInSelectPatient() {
      //signature
      final LocalVariable
            screenId = program.newLocalVariable("screenId", T_INT4),
            sizeofList = program.newLocalVariable("sizeofList", T_INT4),
            msgList = program.newLocalVariable("msgList", DATA_MSG_GROUP_ARRAY),
            pscrData = program.newLocalVariable("pscrData", DBASCR_SCR_DATA_PTR_TYPE);
      tpcrInSelectPatient = program.newProcedure(
                  "tpcrInSelectPatient",
                  Arrays.<LocalVariable>asList(screenId, sizeofList, msgList, pscrData),
                  Arrays.<LocalVariable>asList(pscrData));

      //body
      final LocalVariable num = program.newLocalVariable("num", T_INT4);
      final AssignStmt initializenum = tpcrInSelectPatient.newAssign(num, zero);
      final BranchNode loop_head_condition = tpcrInSelectPatient.newBranch(num.lt(sizeofList));
      final LocalVariable current = program.newLocalVariable("current", DATA_MSG_GROUP);
      final AssignStmt assign_current = tpcrInSelectPatient.newAssign(current, num.join(msgList.join(msg_grp_array_index)));
      final BranchNode check_widgetGroupId = tpcrInSelectPatient.newBranch(current.join(widgetGroupId_field).eq(WG_PATIENT));
      final BranchNode check_widgetId = tpcrInSelectPatient.newBranch(current.join(widgetId_field).eq(W_PATIENT_ID));
      final BranchNode check_length = tpcrInSelectPatient.newBranch(((current.join(value_field)).join(strlen)).gt(DBA_PATIENT_ID_LEN));
      final AssignStmt flag_error = tpcrInSelectPatient.newAssign(error_has_occurred, program.trueLiteral());
      final AssignStmt assign__id_patient = tpcrInSelectPatient.newAssign(dba_patient_type__id_patient, dba_patient_type__id_patient.override(
                  (pscrData.join(scrCrtPatientData)).product(current.join(value_field))));
      final AssignStmt loop_end_num_increment = tpcrInSelectPatient.newAssign(num, num.plus(one));
      final CallStmt call_eventsTPCRSelectPatient = tpcrInSelectPatient.newCall(
                  eventsTPCRSelectPatient,
                  Arrays.<ForgeExpression>asList(screenId, pscrData),
                  Arrays.<ForgeVariable>asList());

      //linkups
      initializenum.setEntry();
      initializenum.setNext(loop_head_condition);
      loop_head_condition.setThen(assign_current);
            assign_current.setNext(check_widgetGroupId);
            check_widgetGroupId.setThen(check_widgetId);
                  check_widgetId.setThen(check_length);
                        check_length.setThen(flag_error);
                              flag_error.setNext(loop_end_num_increment);
                        check_length.setElse(assign__id_patient);
                              assign__id_patient.setNext(loop_end_num_increment);
                        check_widgetId.setElse(loop_end_num_increment);
            check_widgetGroupId.setElse(loop_end_num_increment);
            loop_end_num_increment.setNext(loop_head_condition);
      loop_head_condition.setElse(call_eventsTPCRSelectPatient);
      call_eventsTPCRSelectPatient.setNext(tpcrInSelectPatient.exit());
}
```

Forge

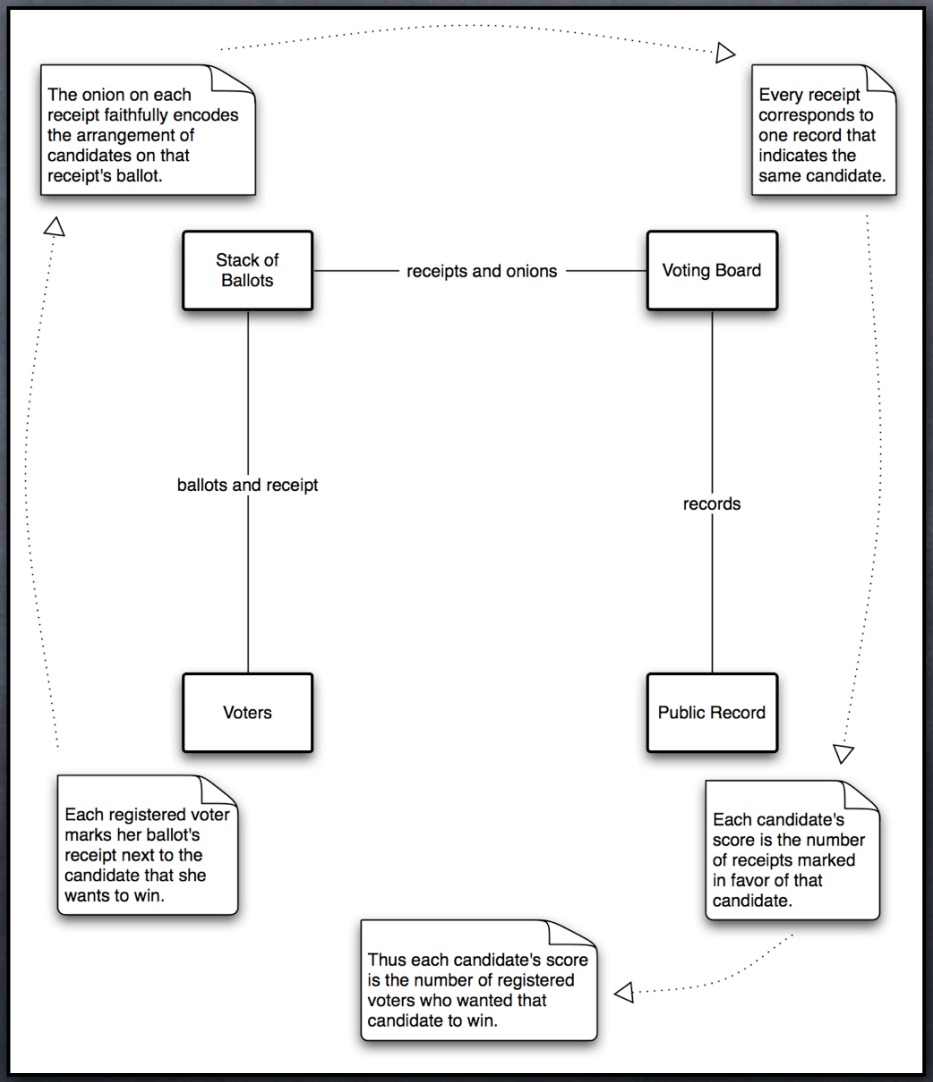
```
proc tpcrInSelectPatient (screenId, sizeofList, msgList, pscrData) : (pscrData) {
  Node55: num := 0 goto Node56
  Node56: if (num < sizeofList) then Node57 else Node64
  Node57: current := (num . (msgList . msg_grp_array_index)) goto Node58
  Node58: if ((current . widgetGroupId_field) = WG_PATIENT) then Node59 else Node63
  Node59: if ((current . widgetId_field) = W_PATIENT_ID) then Node60 else Node63
  Node60: if (((current . value_field) . strlen) > DBA_PATIENT_ID_LEN) then Node61 else Node62
  Node61: error_has_occurred := true goto Node63
  Node63: num := (num plus 1) goto Node56
  Node62: dba_patient_type__id_patient := (dba_patient_type__id_patient ++
          ((pscrData . scrCrtPatientData) -> (current . value_field))) goto Node63
  Node64: eventsTPCRSelectPatient(screenId, pscrData) : () goto Node54
  Node54: terminate
}
```



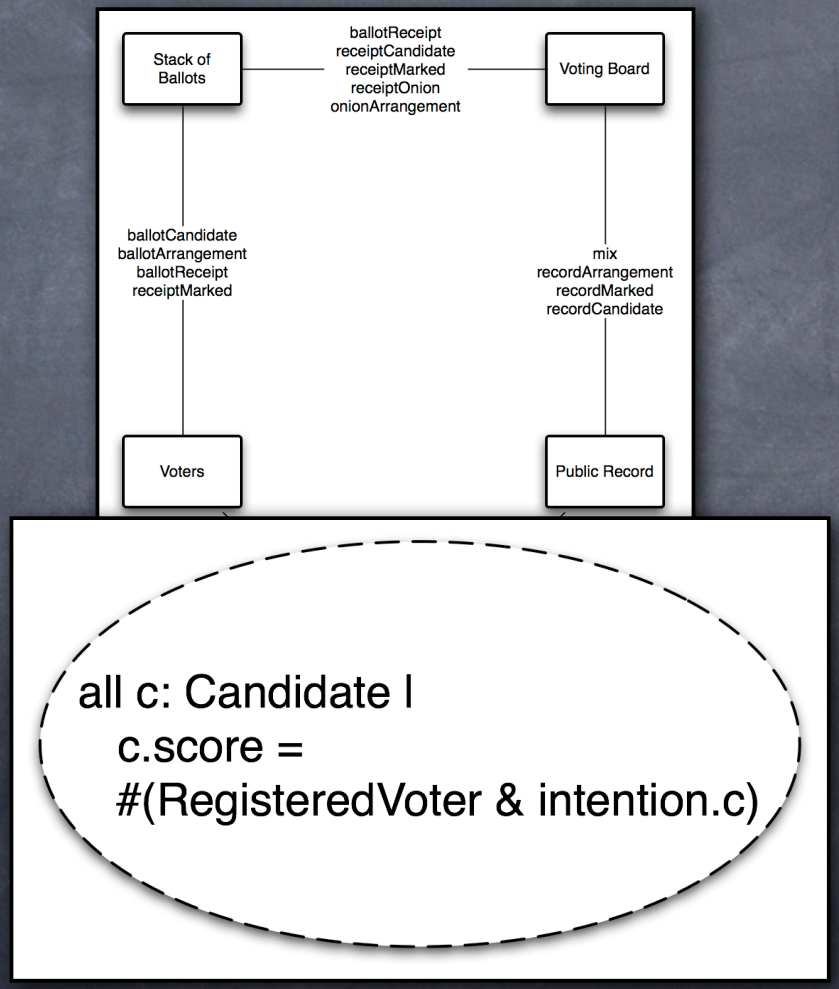
Fidelity


stated as property on...
block
module
component
system
context
context
system
component
module
recast as properties on...
fidelity requirement, designations, problem diagram
component assumptions
requirement progression
mathematical theorems
cryptographic proofs

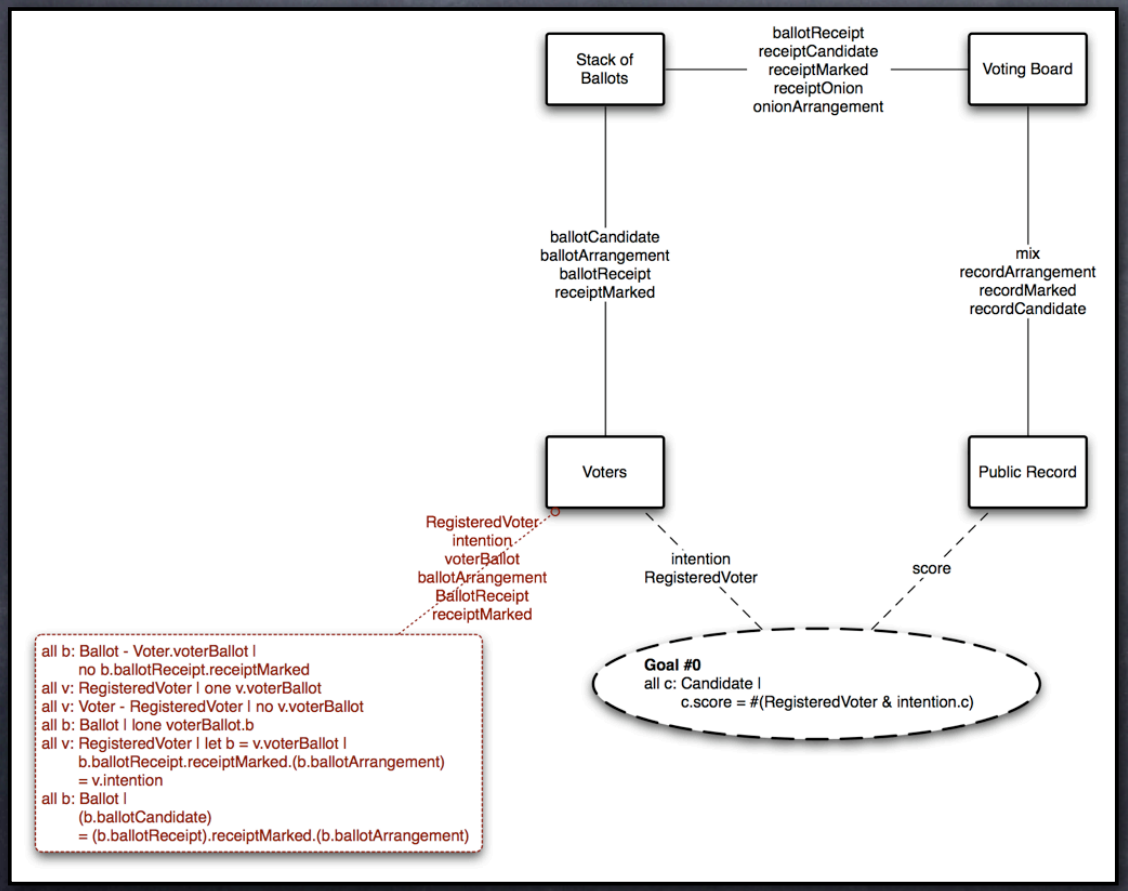
Fidelity


The onion on each receipt faithfully encodes the arrangement of candidates on that receipt's ballot.
Every receipt corresponds to one record that indicates the same candidate.
Stack of Ballots
receipts and onions
Voting Board
ballots and receipt
records
Voters
Public Record
Each registered voter marks her ballot's receipt next to the candidate that she wants to win.
Each candidate's score is the number of receipts marked in favor of that candidate.
Thus each candidate's score is the number of registered voters who wanted that candidate to win.

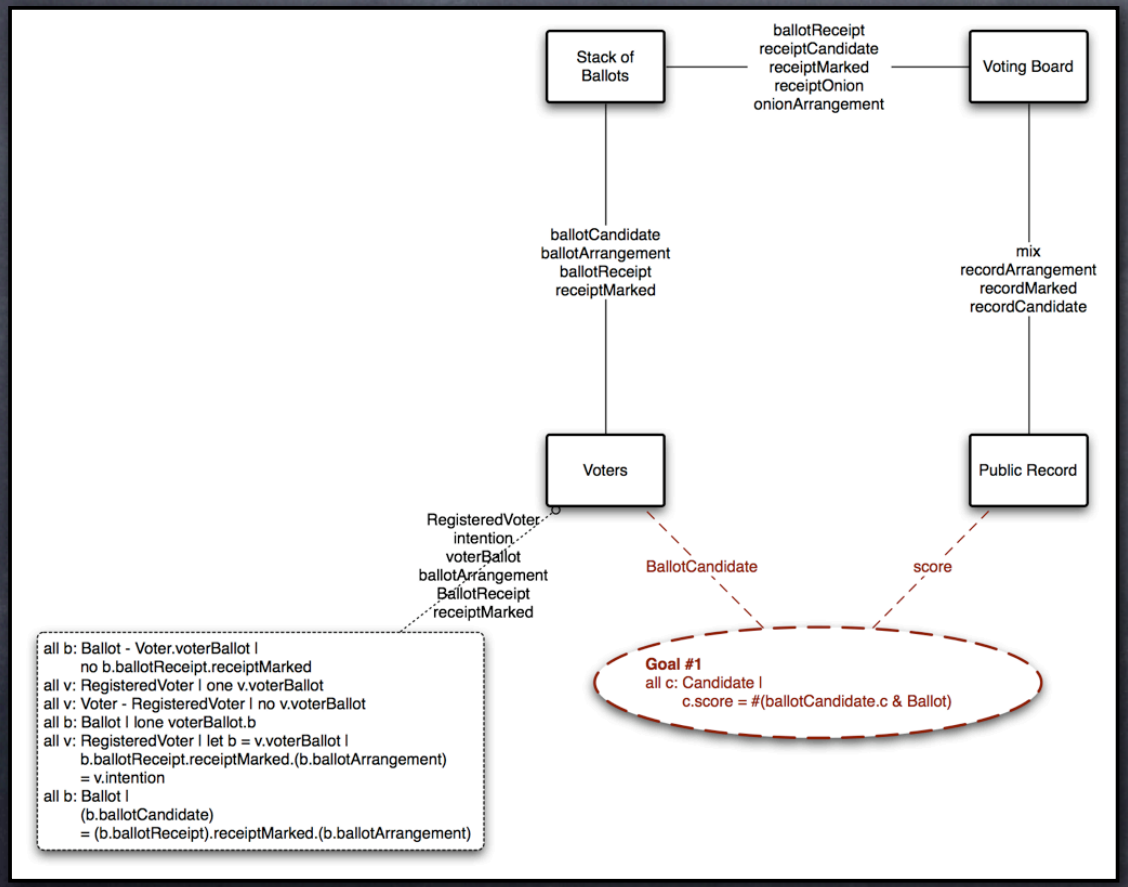
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
Voters
Public Record
all c: Candidate |
c.score =
#(RegisteredVoter & intention.c)

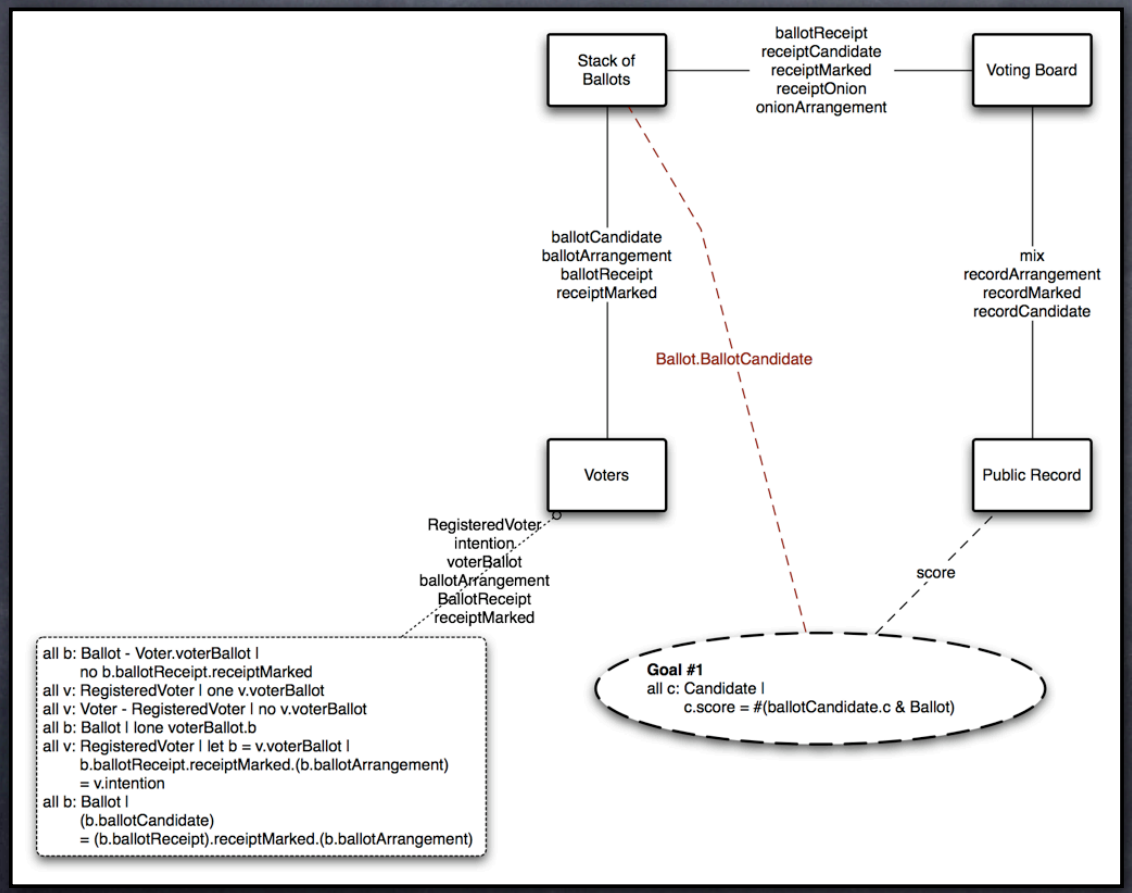
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
intention
RegisteredVoter
score
Goal #0
all c: Candidate |
c.score = #(RegisteredVoter & intention.c)
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)

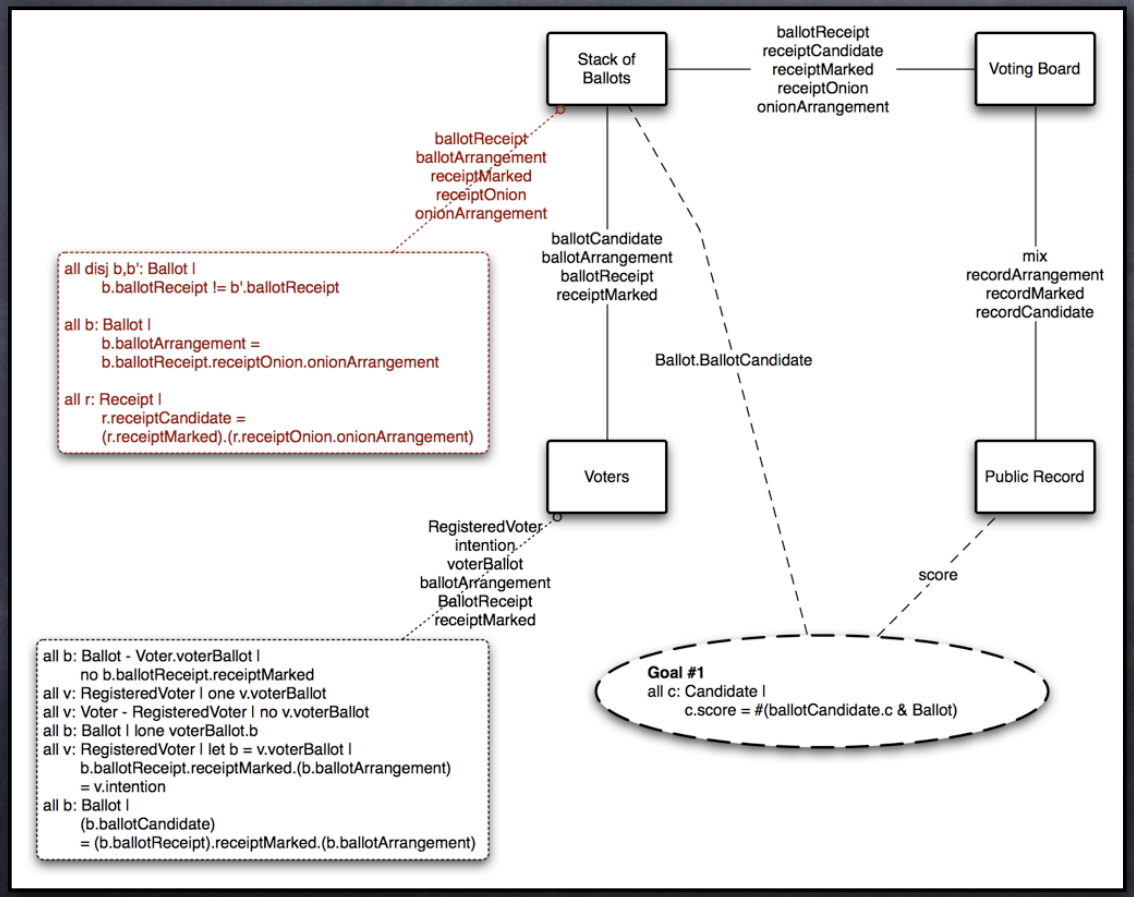
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
BallotCandidate
score
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
Goal #1
all c: Candidate |
c.score = #(ballotCandidate.c & Ballot)

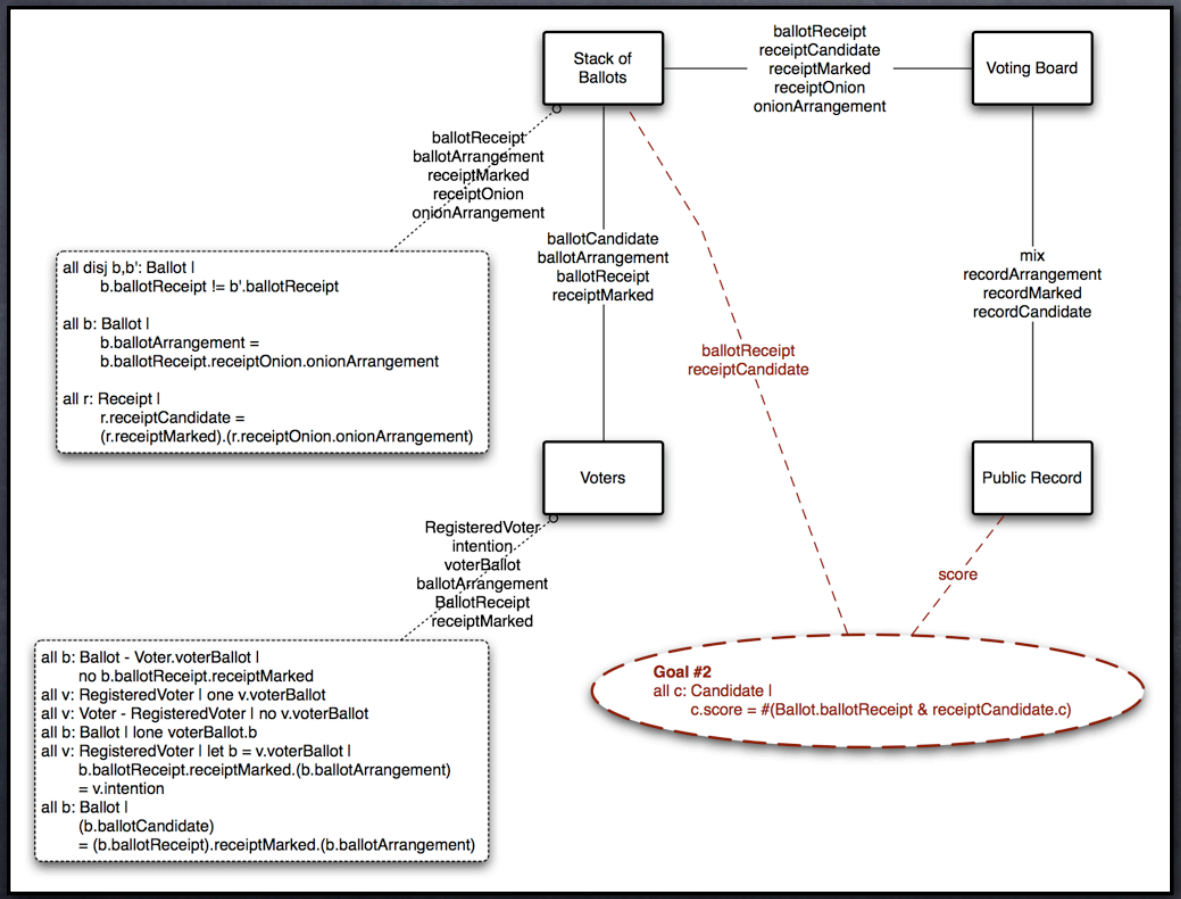
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
Ballot.BallotCandidate
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
score
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
Goal #1
all c: Candidate |
c.score = #(ballotCandidate.c & Ballot)

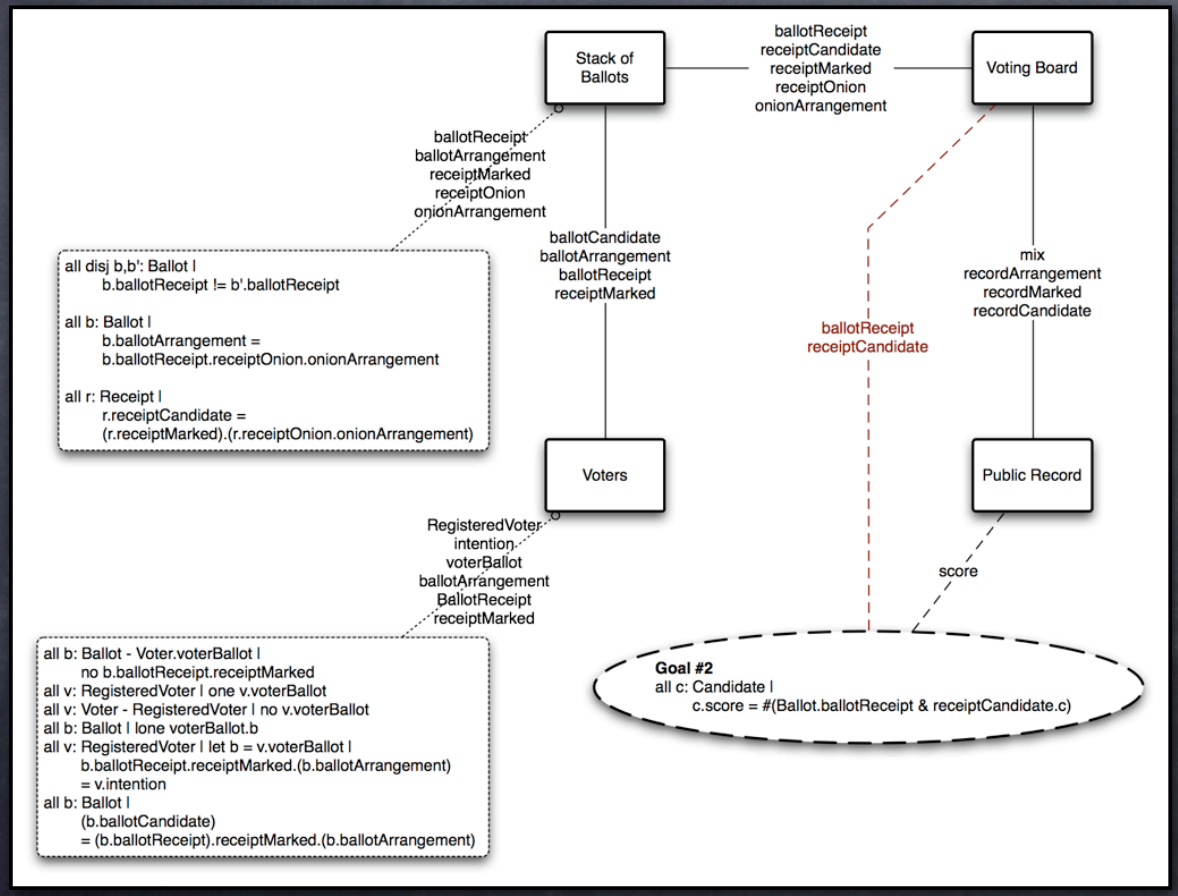
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotReceipt
ballotArrangement
receiptMarked
receiptOnion
onionArrangement
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
Ballot.BallotCandidate
all disj b,b': Ballot |
b.ballotReceipt != b'.ballotReceipt
all b: Ballot |
b.ballotArrangement =
b.ballotReceipt.receiptOnion.onionArrangement
all r: Receipt |
r.receiptCandidate =
(r.receiptMarked).(r.receiptOnion.onionArrangement)
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
score
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
Goal #1
all c: Candidate |
c.score = #(ballotCandidate.c & Ballot)

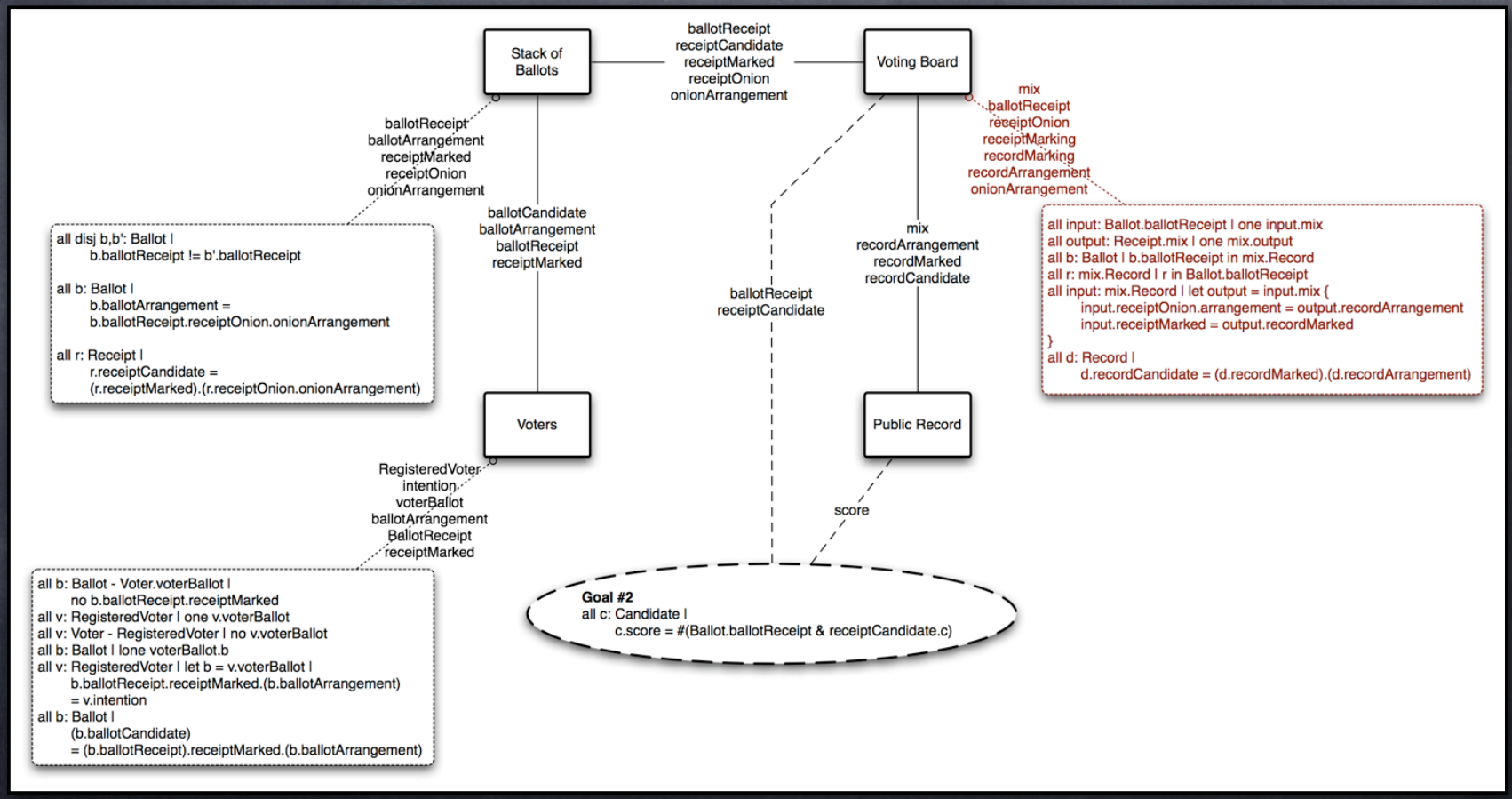
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotReceipt
ballotArrangement
receiptMarked
receiptOnion
onionArrangement
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
ballotReceipt
receiptCandidate
all disj b,b': Ballot |
b.ballotReceipt != b'.ballotReceipt
all b: Ballot |
b.ballotArrangement =
b.ballotReceipt.receiptOnion.onionArrangement
all r: Receipt |
r.receiptCandidate =
(r.receiptMarked).(r.receiptOnion.onionArrangement)
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
score
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
Goal #2
all c: Candidate |
c.score = #(Ballot.ballotReceipt & receiptCandidate.c)

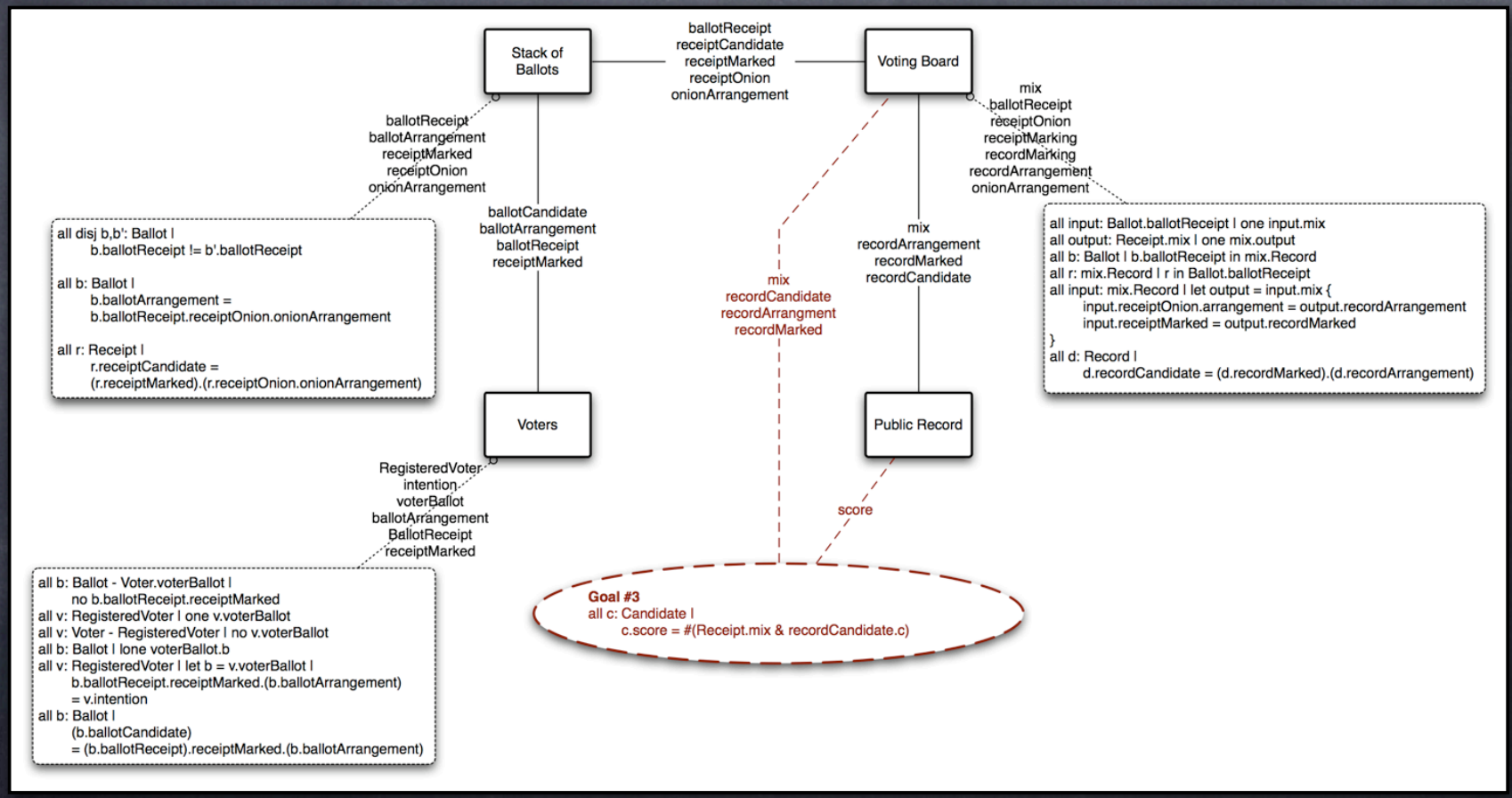
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotReceipt
ballotArrangement
receiptMarked
receiptOnion
onionArrangement
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
ballotReceipt
receiptCandidate
all disj b,b': Ballot |
b.ballotReceipt != b'.ballotReceipt
all b: Ballot |
b.ballotArrangement =
b.ballotReceipt.receiptOnion.onionArrangement
all r: Receipt |
r.receiptCandidate =
(r.receiptMarked).(r.receiptOnion.onionArrangement)
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
score
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
Goal #2
all c: Candidate |
c.score = #(Ballot.ballotReceipt & receiptCandidate.c)

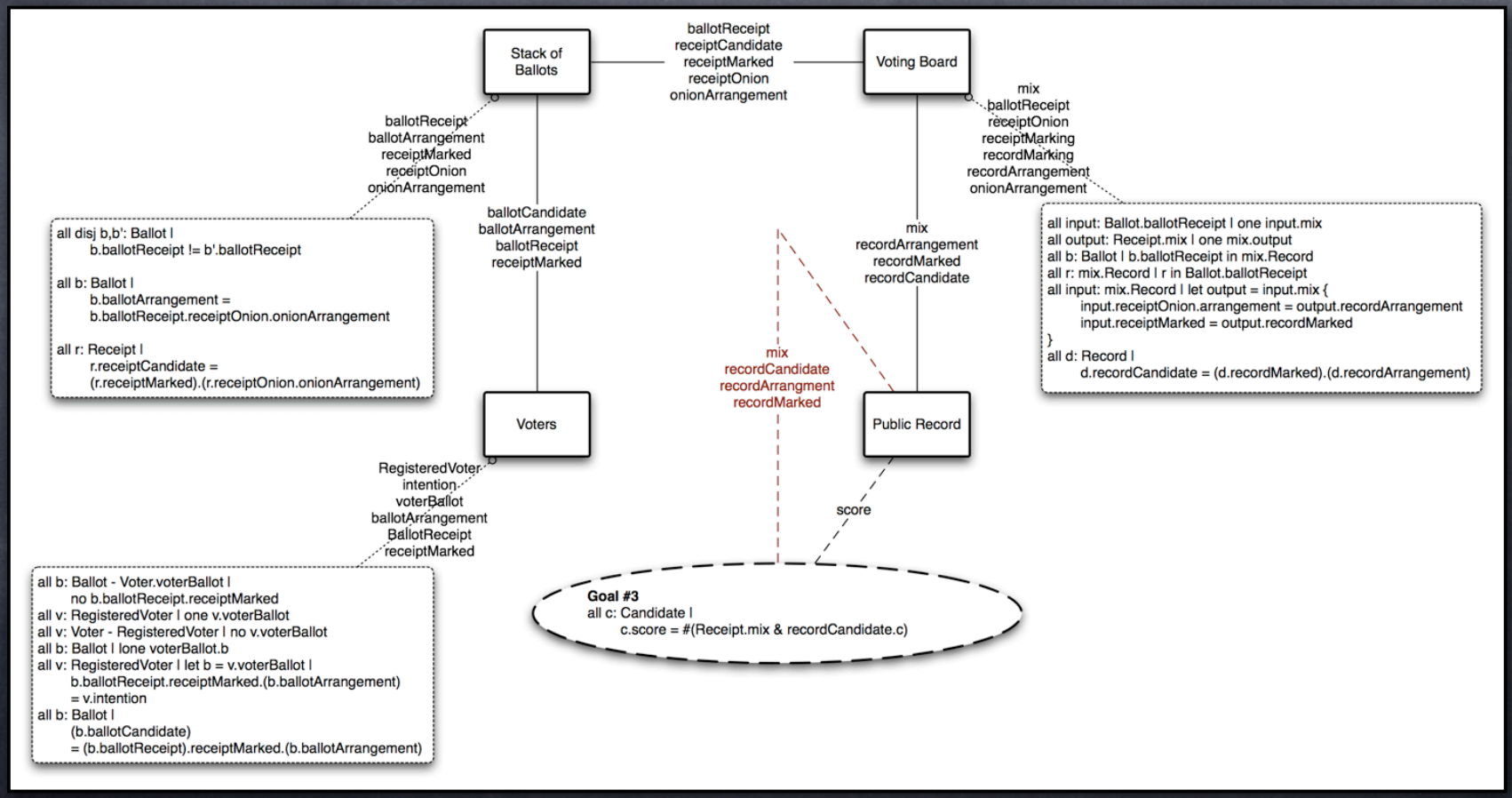
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotReceipt
ballotArrangement
receiptMarked
receiptOnion
onionArrangement
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
ballotReceipt
receiptOnion
receiptMarking
recordMarking
recordArrangement
onionArrangement
mix
recordArrangement
recordMarked
recordCandidate
ballotReceipt
receiptCandidate
all disj b,b': Ballot |
b.ballotReceipt != b'.ballotReceipt
all b: Ballot |
b.ballotArrangement =
b.ballotReceipt.receiptOnion.onionArrangement
all r: Receipt |
r.receiptCandidate =
(r.receiptMarked).(r.receiptOnion.onionArrangement)
all input: Ballot.ballotReceipt | one input.mix
all output: Receipt.mix | one mix.output
all b: Ballot | b.ballotReceipt in mix.Record
all r: mix.Record | r in Ballot.ballotReceipt
all input: mix.Record | let output = input.mix {
input.receiptOnion.arrangement = output.recordArrangement
input.receiptMarked = output.recordMarked
}
all d: Record |
d.recordCandidate = (d.recordMarked).(d.recordArrangement)
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
score
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
Goal #2
all c: Candidate |
c.score = #(Ballot.ballotReceipt & receiptCandidate.c)

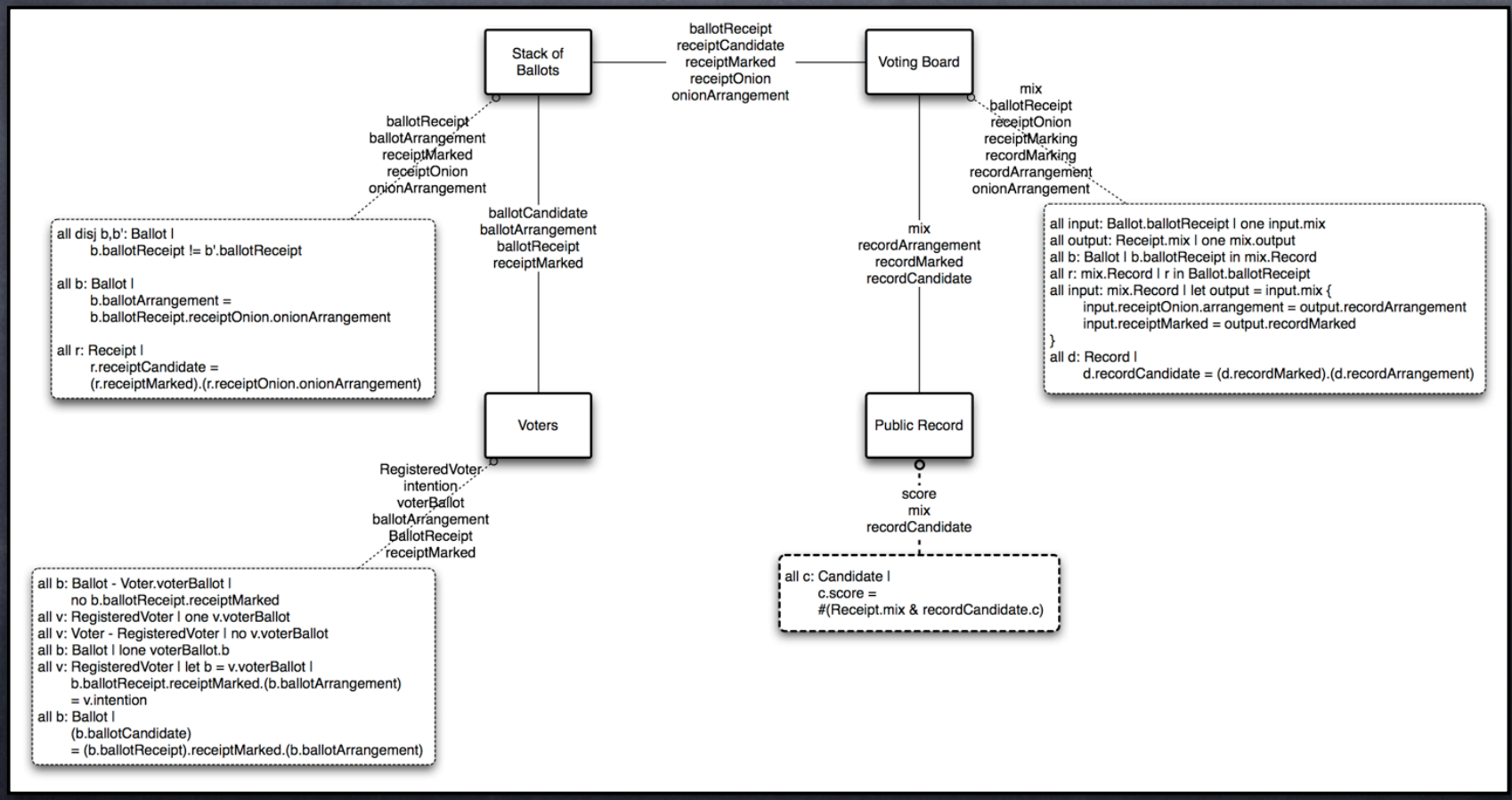
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotReceipt
ballotArrangement
receiptMarked
receiptOnion
onionArrangement
mix
ballotReceipt
receiptOnion
receiptMarking
recordMarking
recordArrangement
onionArrangement
all disj b,b': Ballot |
b.ballotReceipt != b'.ballotReceipt
all b: Ballot |
b.ballotArrangement =
b.ballotReceipt.receiptOnion.onionArrangement
all r: Receipt |
r.receiptCandidate =
(r.receiptMarked).(r.receiptOnion.onionArrangement)
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
mix
recordCandidate
recordArrangment
recordMarked
all input: Ballot.ballotReceipt | one input.mix
all output: Receipt.mix | one mix.output
all b: Ballot | b.ballotReceipt in mix.Record
all r: mix.Record | r in Ballot.ballotReceipt
all input: mix.Record | let output = input.mix {
input.receiptOnion.arrangement = output.recordArrangement
input.receiptMarked = output.recordMarked
}
all d: Record |
d.recordCandidate = (d.recordMarked).(d.recordArrangement)
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
score
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
Goal #3
all c: Candidate |
c.score = #(Receipt.mix & recordCandidate.c)

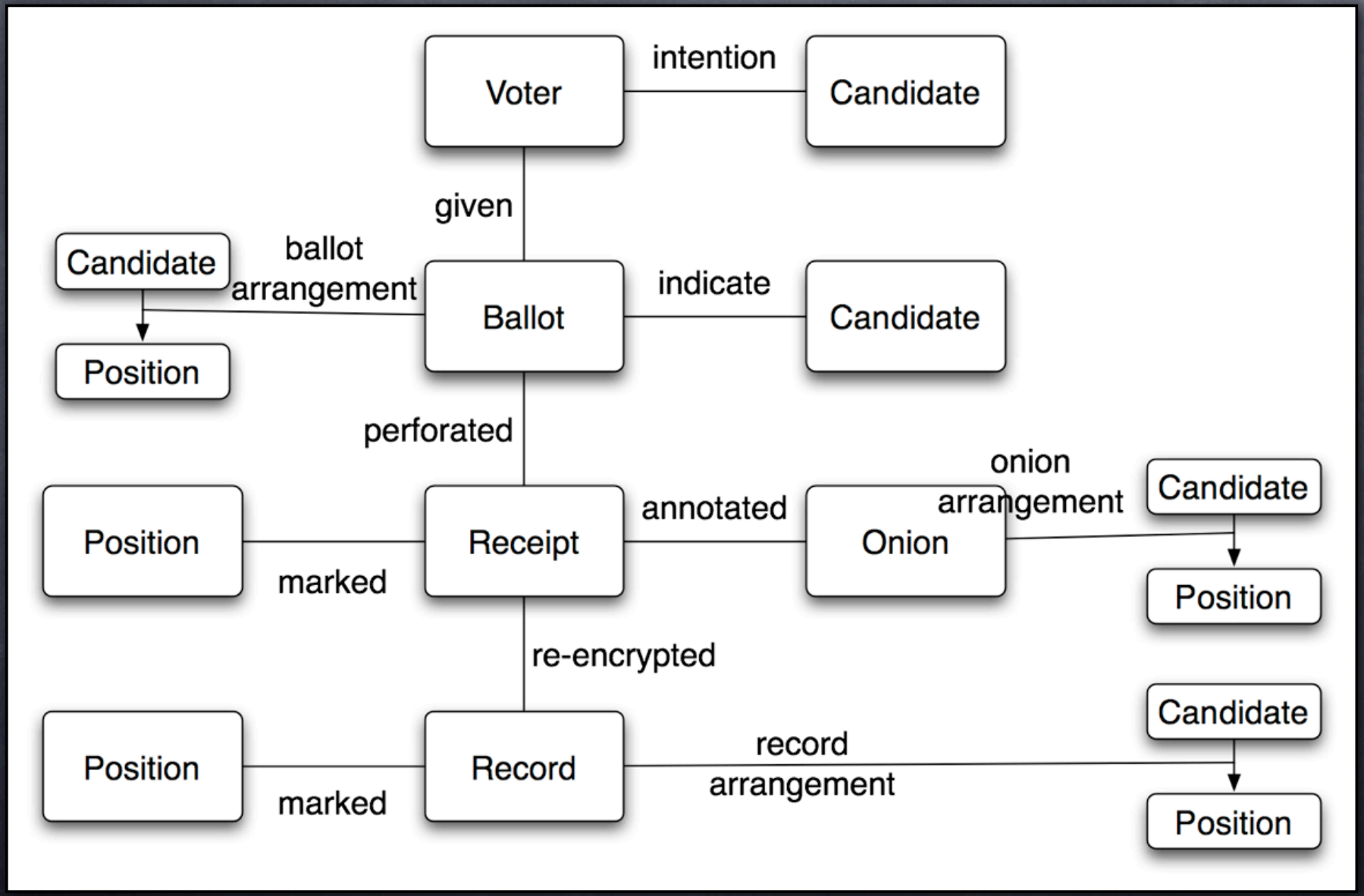
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotReceipt
ballotArrangement
receiptMarked
receiptOnion
onionArrangement
mix
ballotReceipt
receiptOnion
receiptMarking
recordMarking
recordArrangement
onionArrangement
all disj b,b': Ballot |
b.ballotReceipt != b'.ballotReceipt
all b: Ballot |
b.ballotArrangement =
b.ballotReceipt.receiptOnion.onionArrangement
all r: Receipt |
r.receiptCandidate =
(r.receiptMarked).(r.receiptOnion.onionArrangement)
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
all input: Ballot.ballotReceipt | one input.mix
all output: Receipt.mix | one mix.output
all b: Ballot | b.ballotReceipt in mix.Record
all r: mix.Record | r in Ballot.ballotReceipt
all input: mix.Record | let output = input.mix {
input.receiptOnion.arrangement = output.recordArrangement
input.receiptMarked = output.recordMarked
}
all d: Record |
d.recordCandidate = (d.recordMarked).(d.recordArrangement)
mix
recordCandidate
recordArrangment
recordMarked
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
score
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
Goal #3
all c: Candidate |
c.score = #(Receipt.mix & recordCandidate.c)

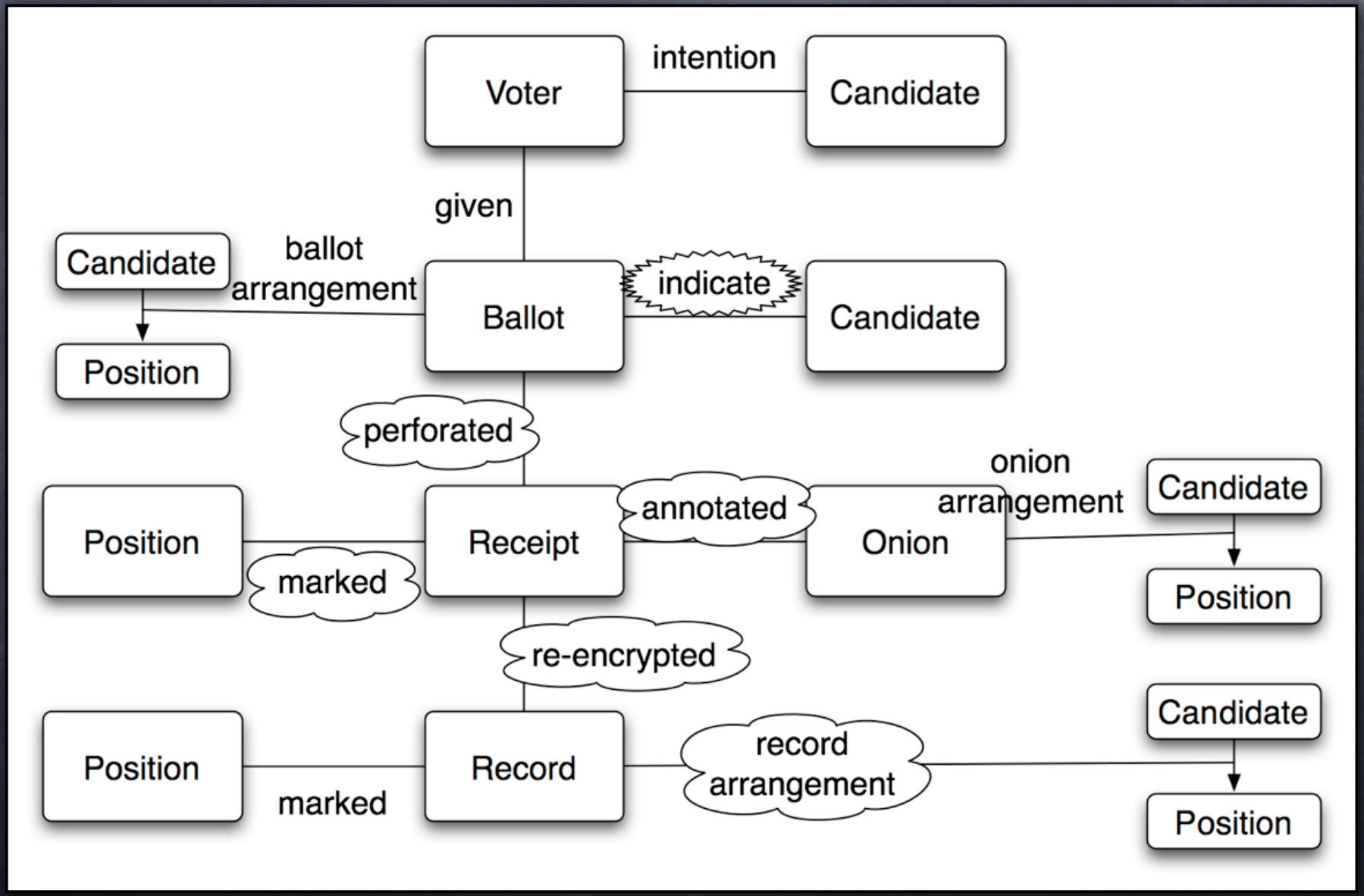
Fidelity


Stack of Ballots
ballotReceipt
receiptCandidate
receiptMarked
receiptOnion
onionArrangement
Voting Board
ballotReceipt
ballotArrangement
receiptMarked
receiptOnion
onionArrangement
mix
ballotReceipt
receiptOnion
receiptMarking
recordMarking
recordArrangement
onionArrangement
all disj b,b': Ballot |
b.ballotReceipt != b'.ballotReceipt
all b: Ballot |
b.ballotArrangement =
b.ballotReceipt.receiptOnion.onionArrangement
all r: Receipt |
r.receiptCandidate =
(r.receiptMarked).(r.receiptOnion.onionArrangement)
ballotCandidate
ballotArrangement
ballotReceipt
receiptMarked
mix
recordArrangement
recordMarked
recordCandidate
all input: Ballot.ballotReceipt | one input.mix
all output: Receipt.mix | one mix.output
all b: Ballot | b.ballotReceipt in mix.Record
all r: mix.Record | r in Ballot.ballotReceipt
all input: mix.Record | let output = input.mix {
input.receiptOnion.arrangement = output.recordArrangement
input.receiptMarked = output.recordMarked
}
all d: Record |
d.recordCandidate = (d.recordMarked).(d.recordArrangement)
Voters
Public Record
RegisteredVoter
intention
voterBallot
ballotArrangement
BallotReceipt
receiptMarked
score
mix
recordCandidate
all b: Ballot - Voter.voterBallot |
no b.ballotReceipt.receiptMarked
all v: RegisteredVoter | one v.voterBallot
all v: Voter - RegisteredVoter | no v.voterBallot
all b: Ballot | lone voterBallot.b
all v: RegisteredVoter | let b = v.voterBallot |
b.ballotReceipt.receiptMarked.(b.ballotArrangement)
= v.intention
all b: Ballot |
(b.ballotCandidate)
= (b.ballotReceipt).receiptMarked.(b.ballotArrangement)
all c: Candidate |
c.score =
#(Receipt.mix & recordCandidate.c)

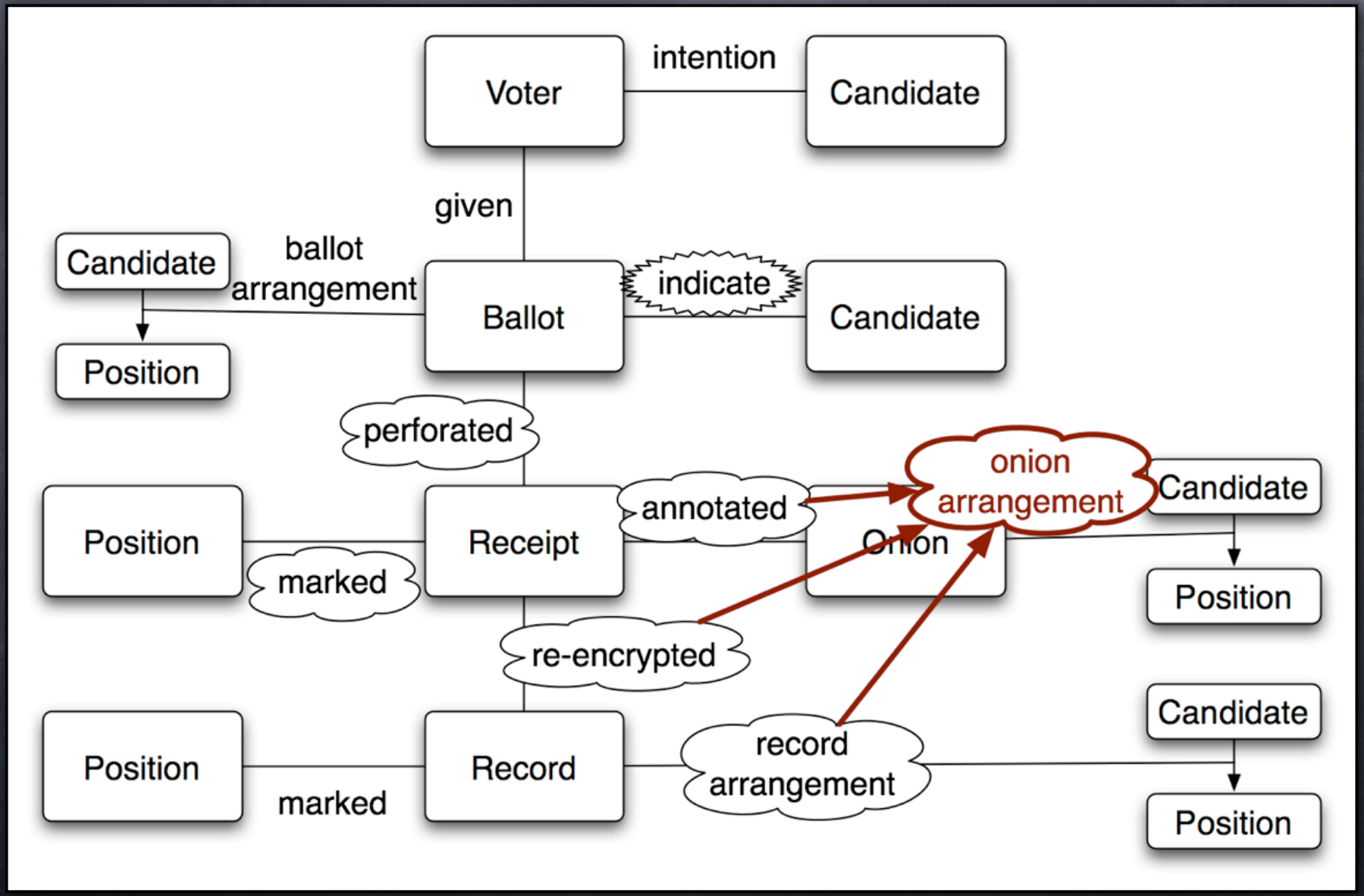
A Secrecy Attack



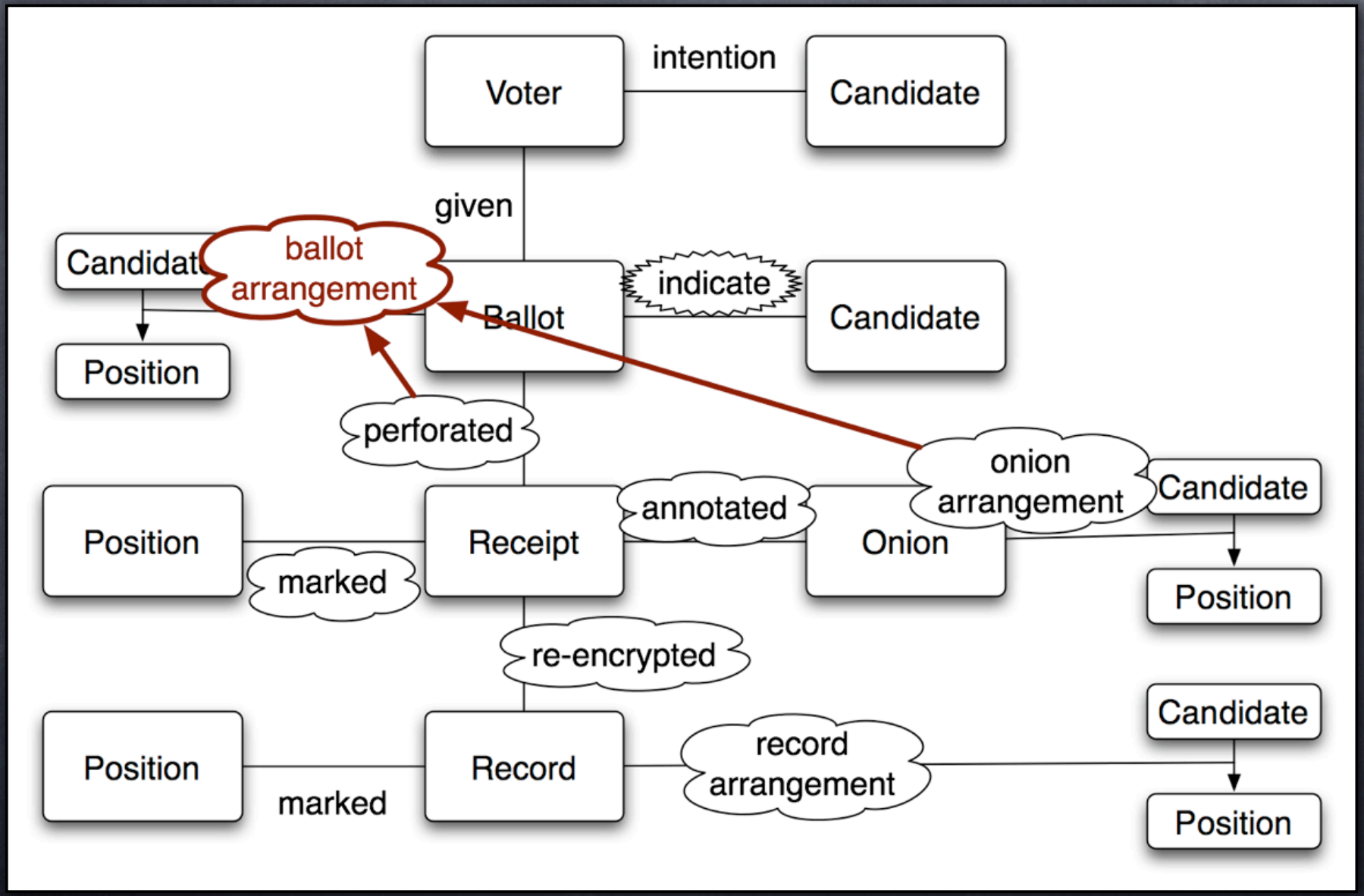
A Secrecy Attack



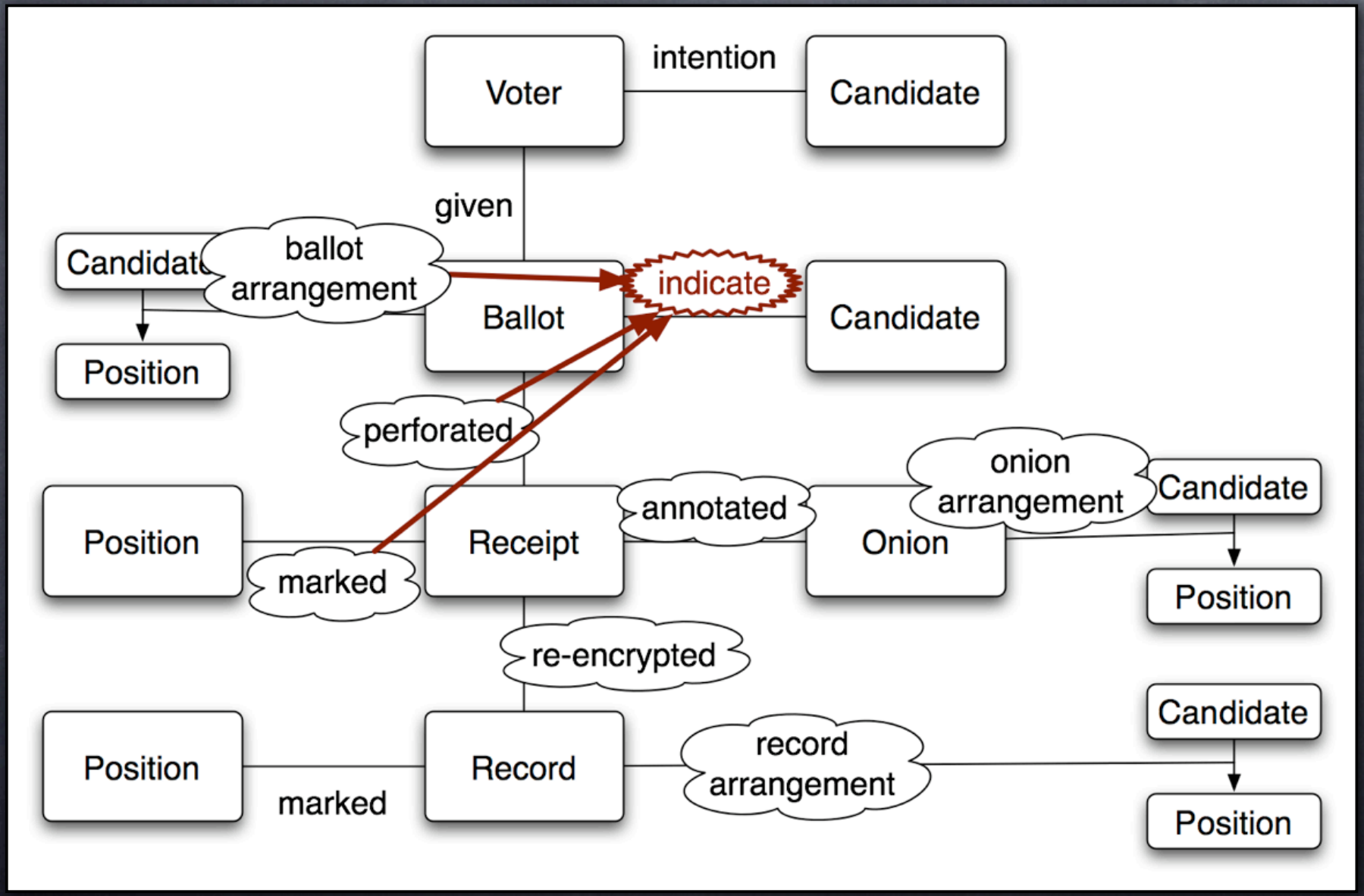
A Secrecy Attack



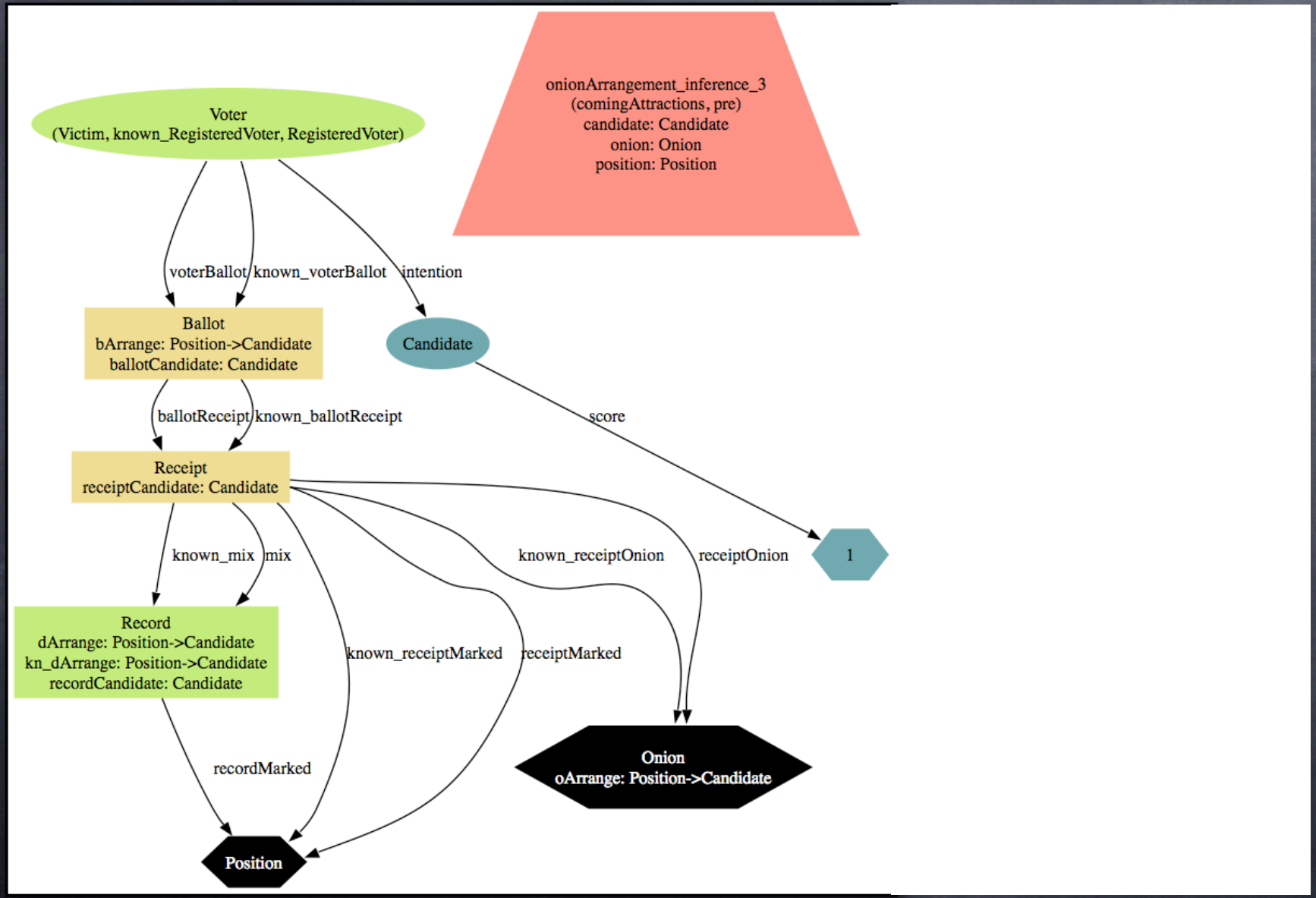
A Secrecy Attack



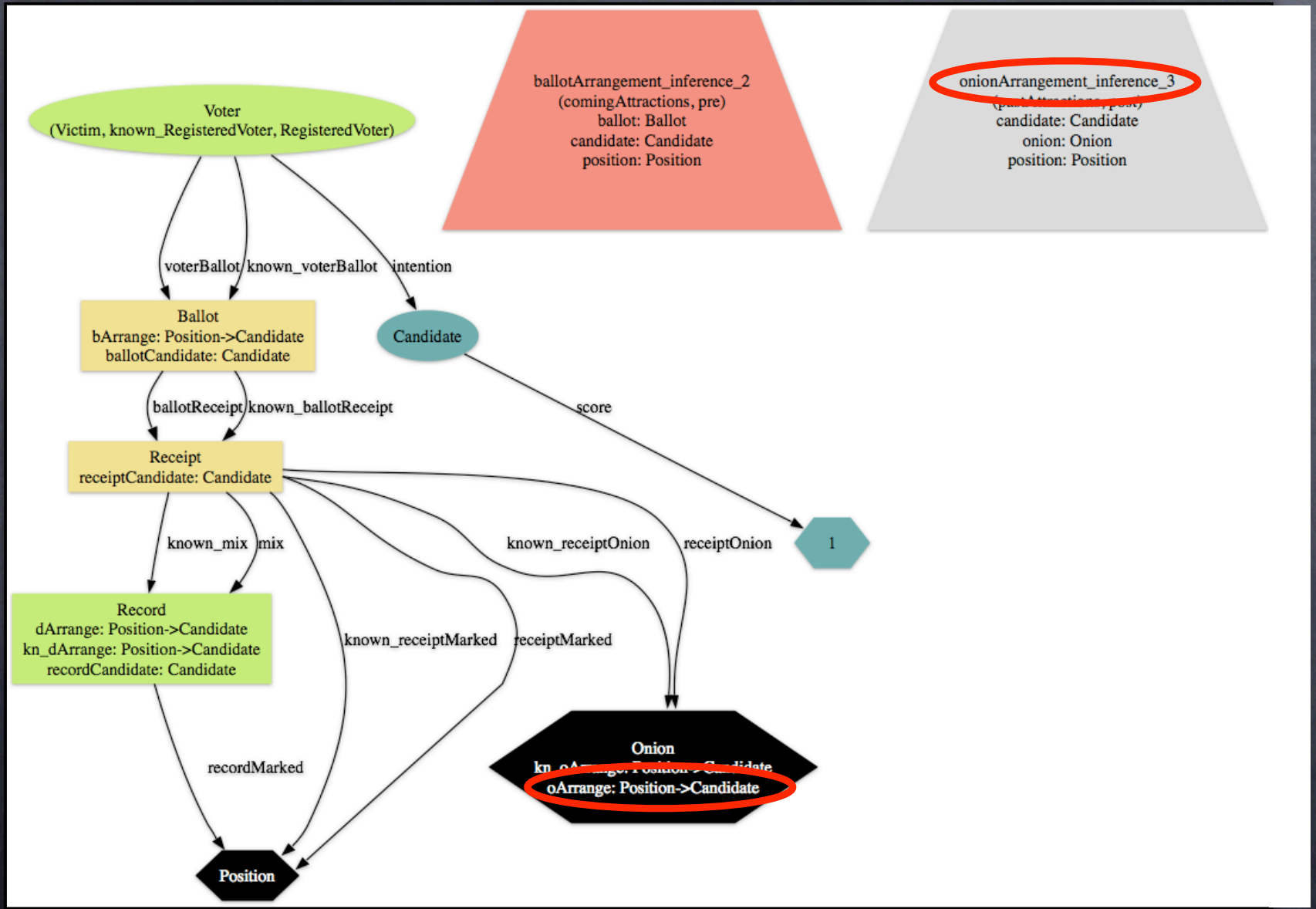
A Secrecy Attack



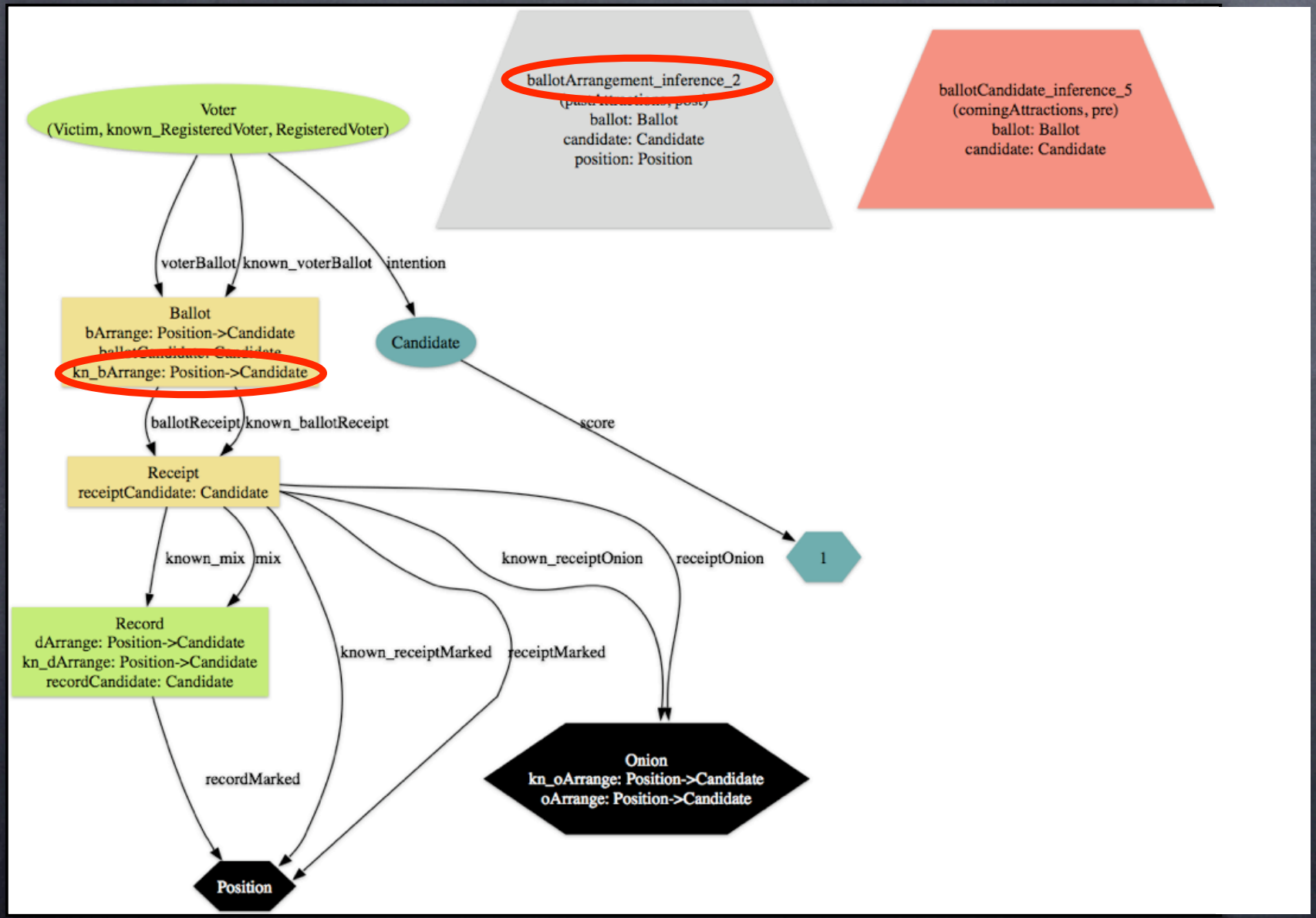
Modeling Secrecy Attacks



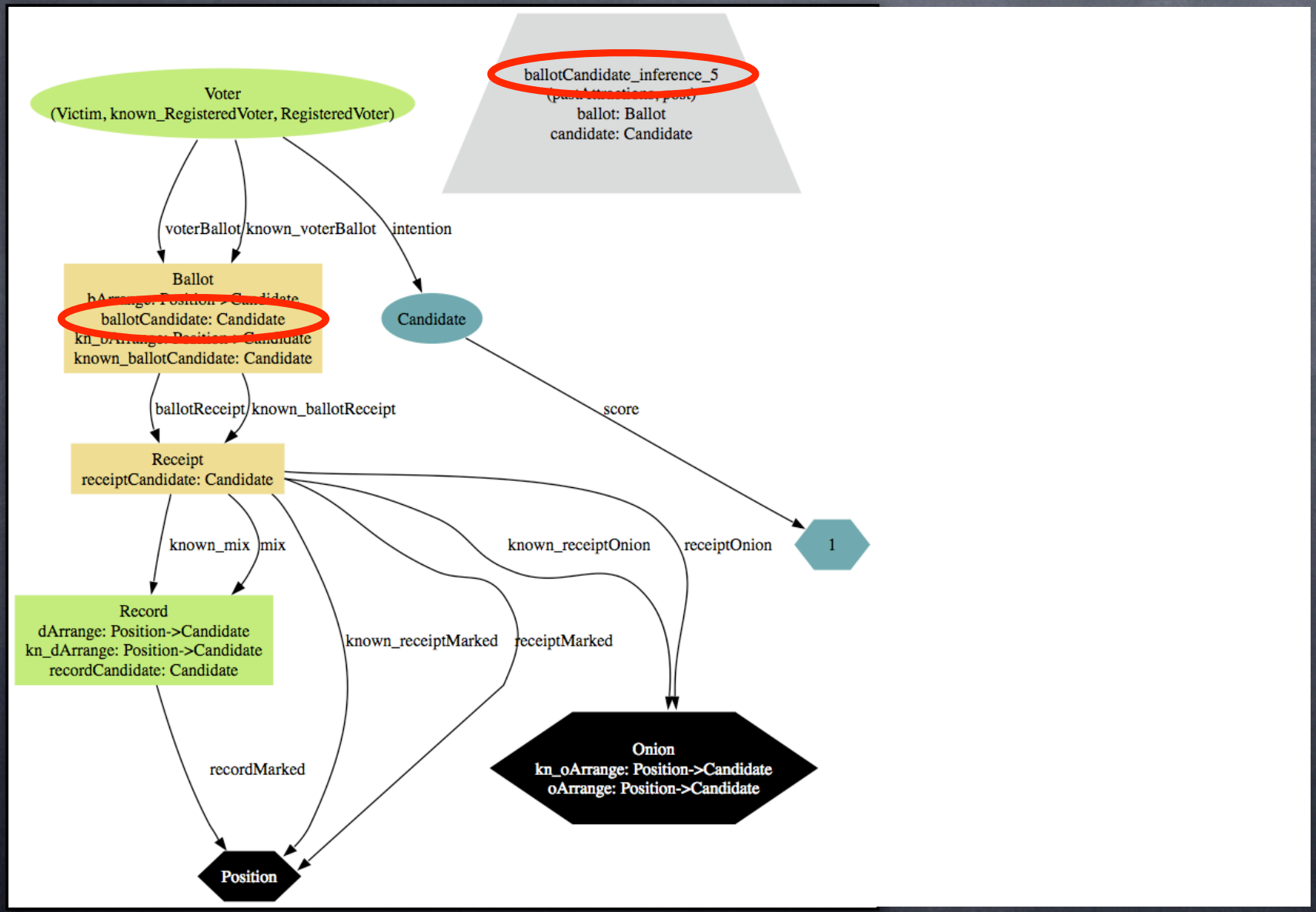
Modeling Secrecy Attacks



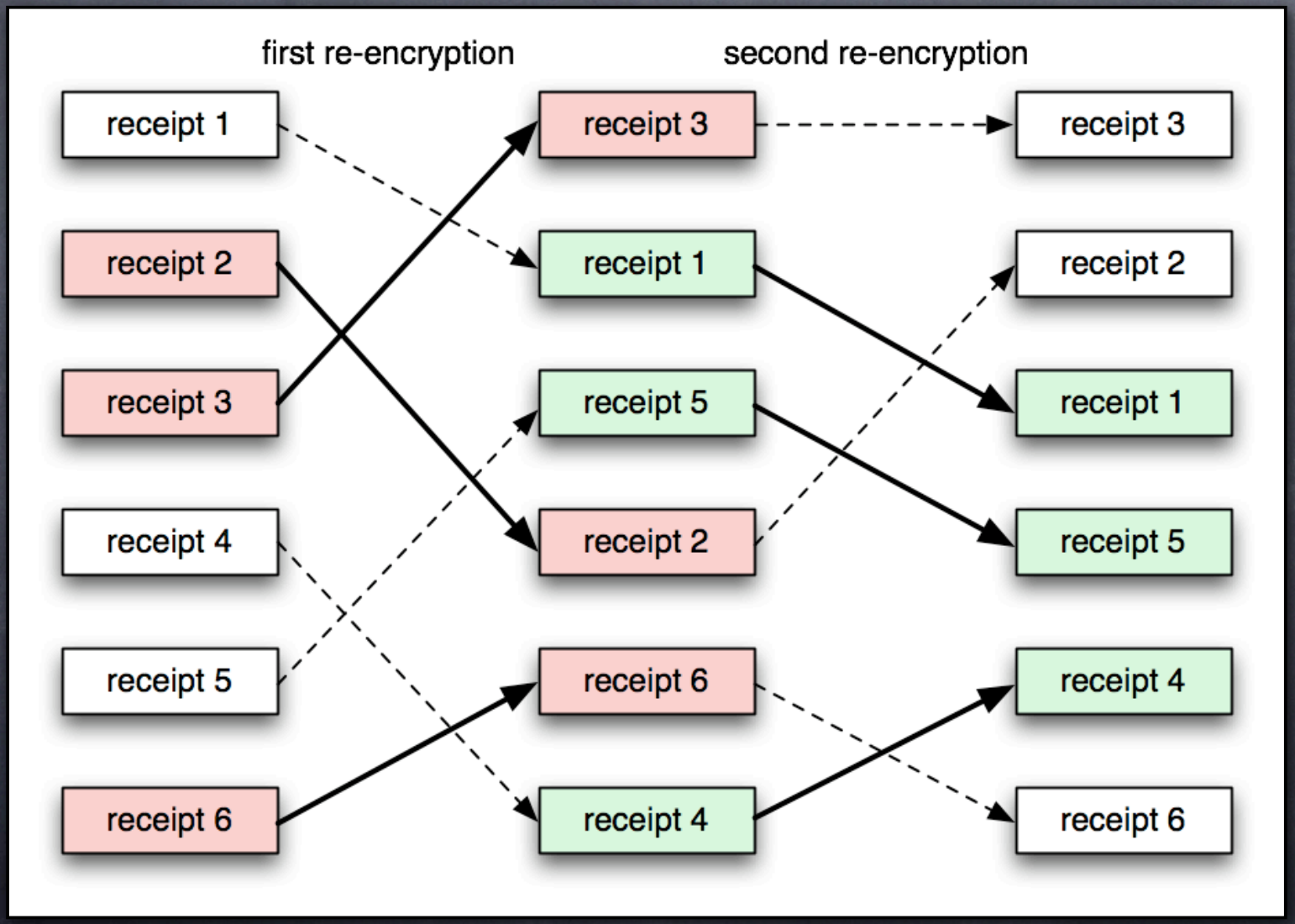
Modeling Secrecy Attacks



Modeling Secrecy Attacks



Auditability



Future Work

Requirement Progression
- tool support (automatic suggestions)

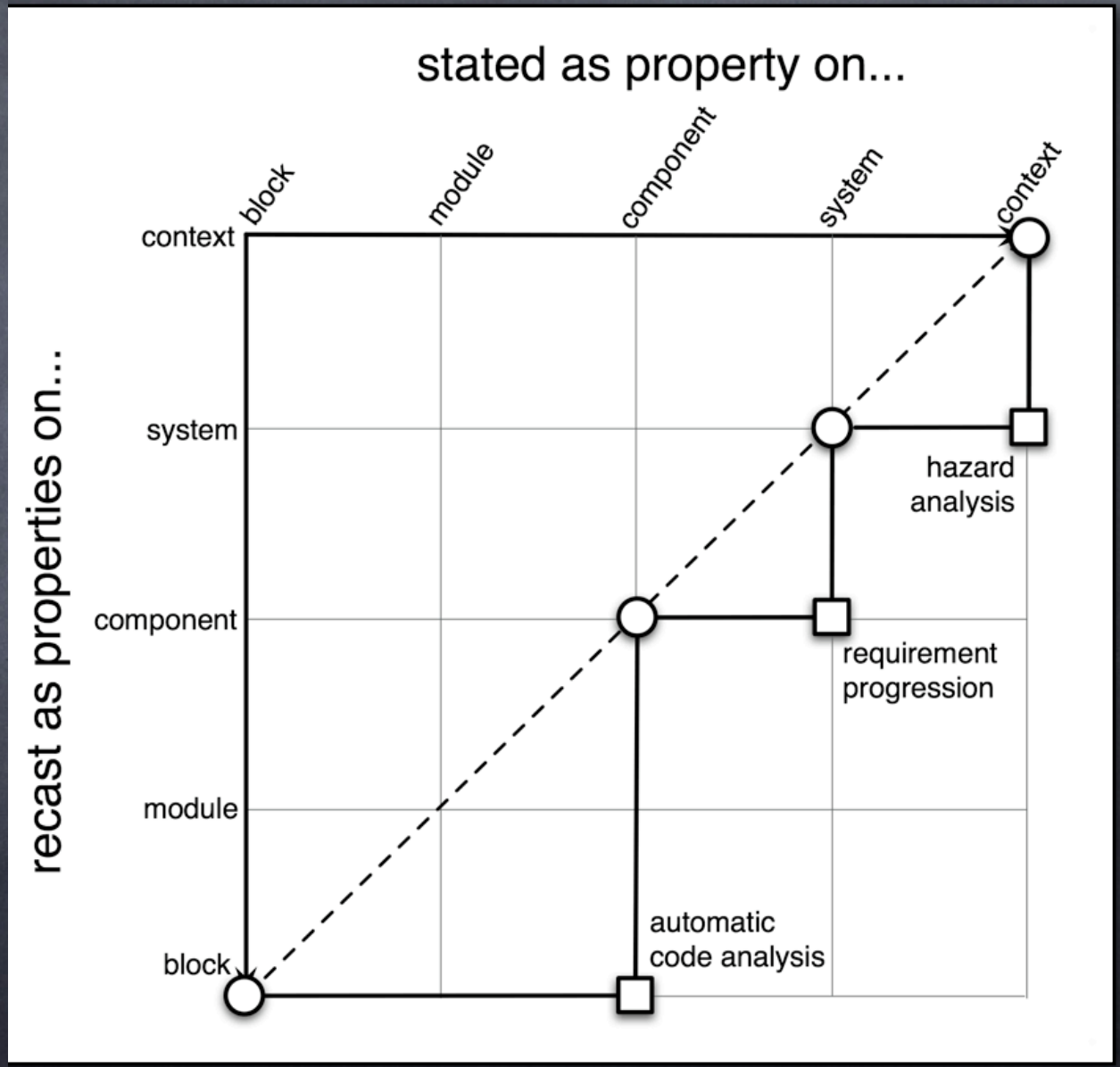
Code Analysis
- lower human cost, better scalability

Human Analysis
- systematic assessment of assumptions

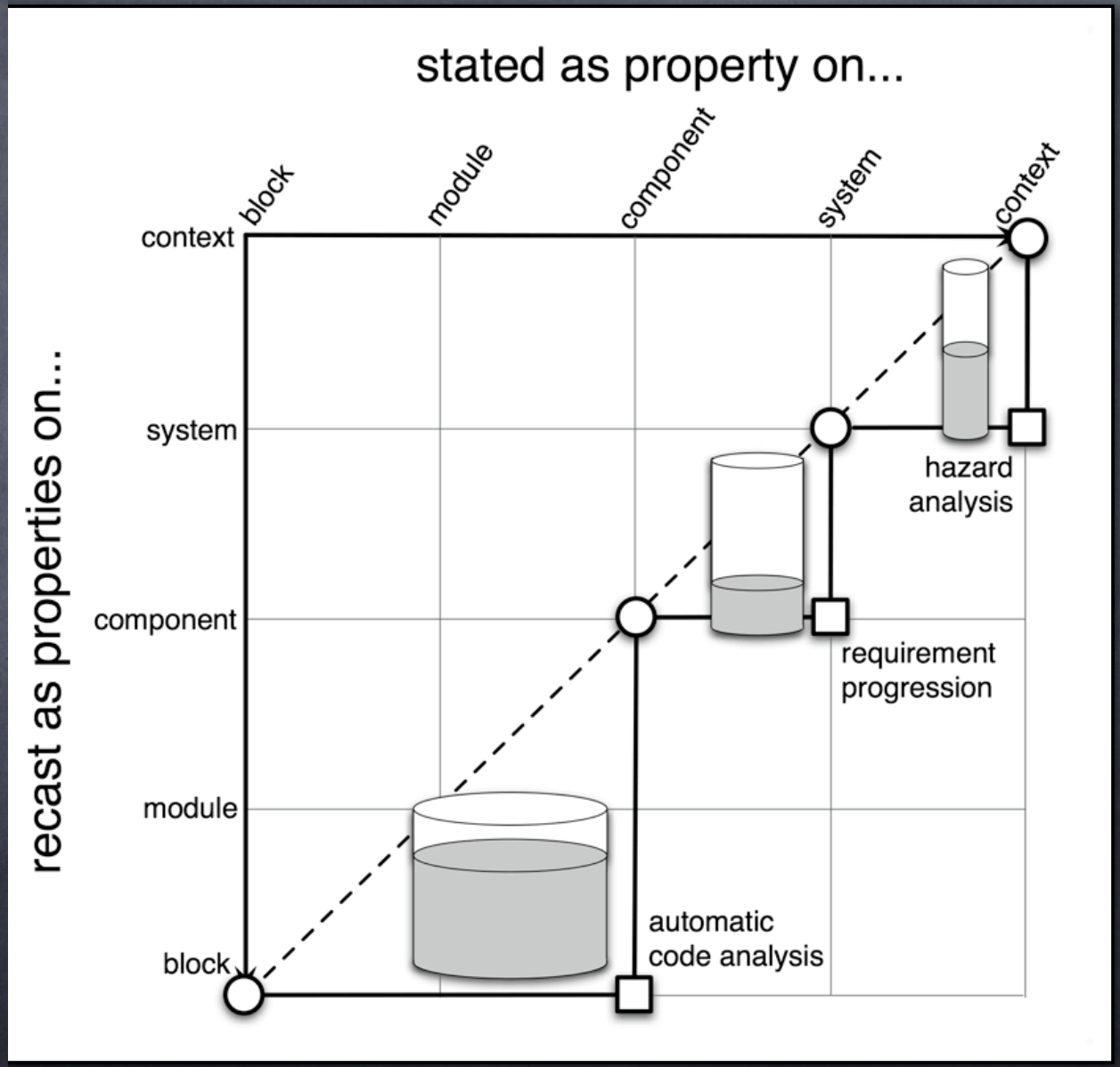
Case Studies
- lightweight, automatic, lower confidence

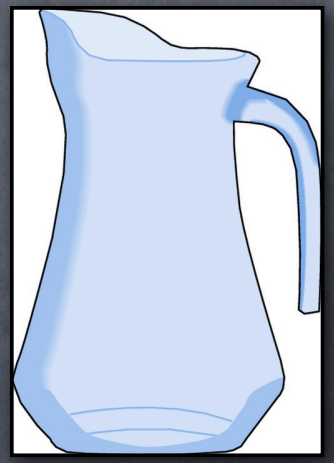
CDAD
- denote confidence, cost

How to invest effort?



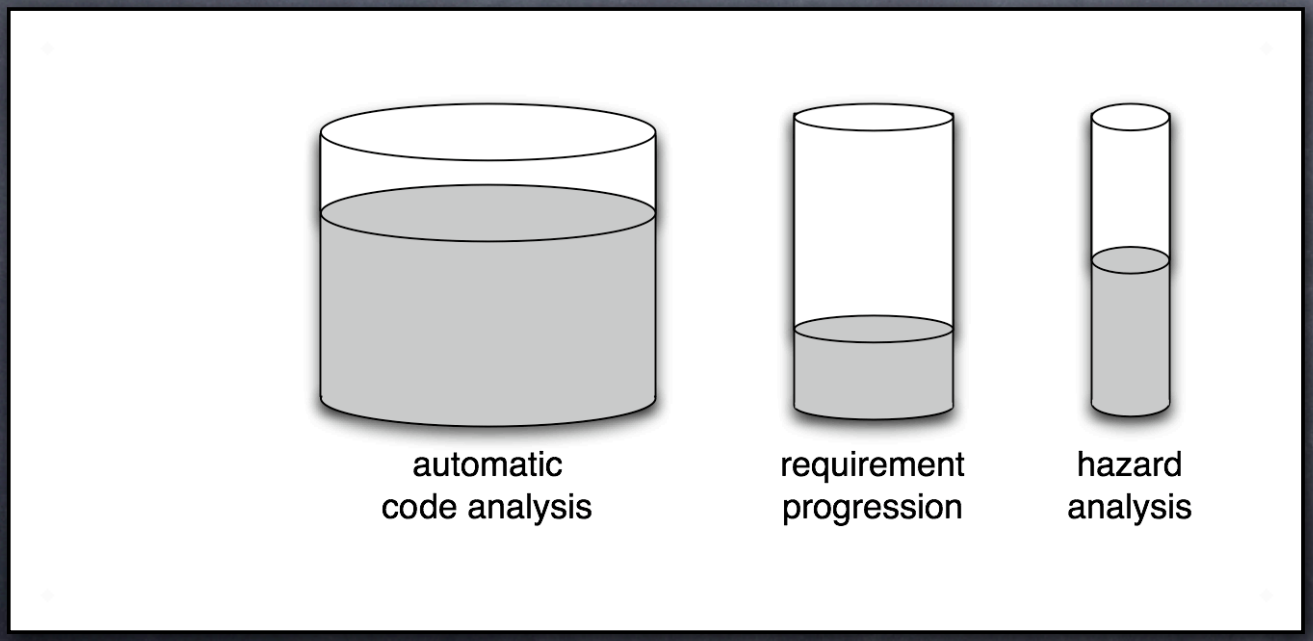
Idea: Water Glass Model

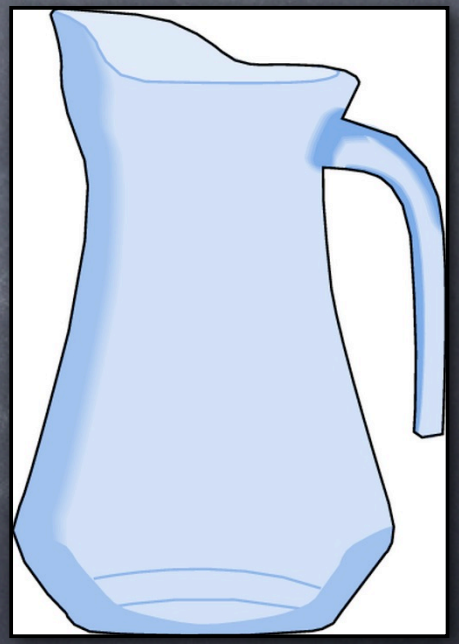




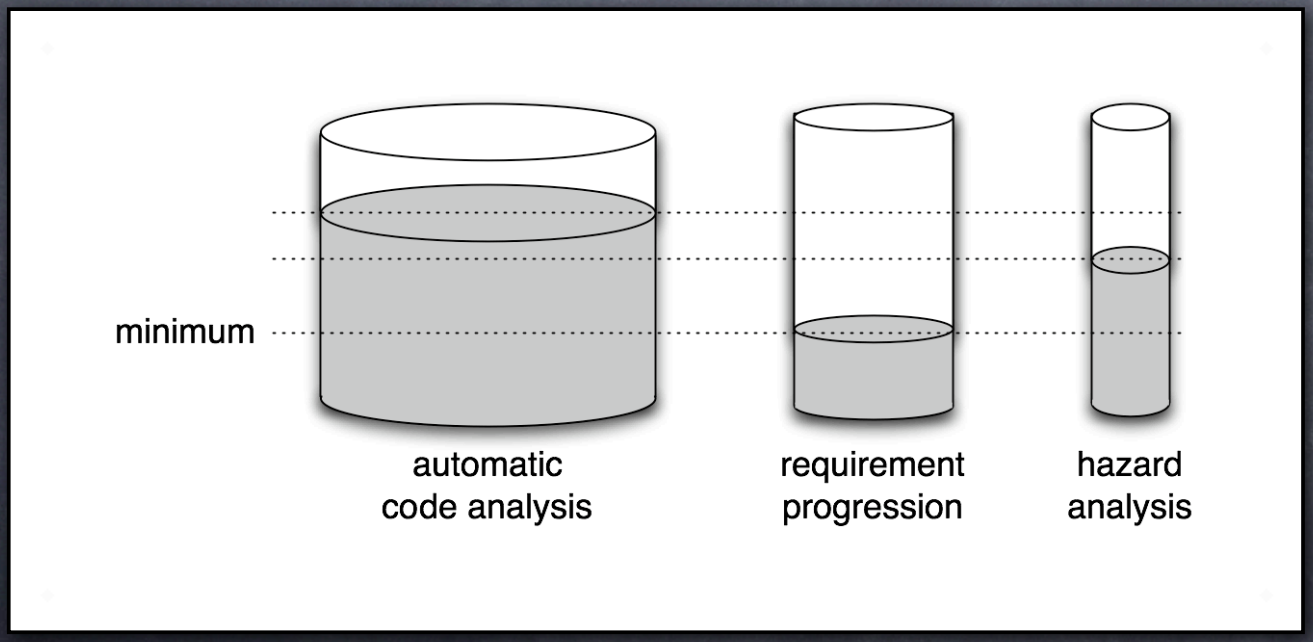
Budget

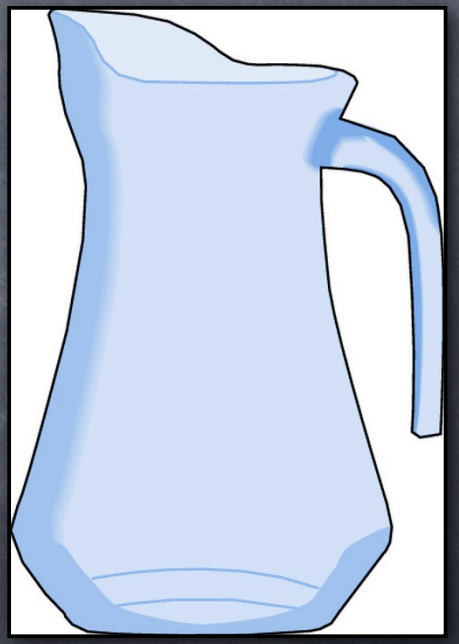
Idea: Water Glass Model



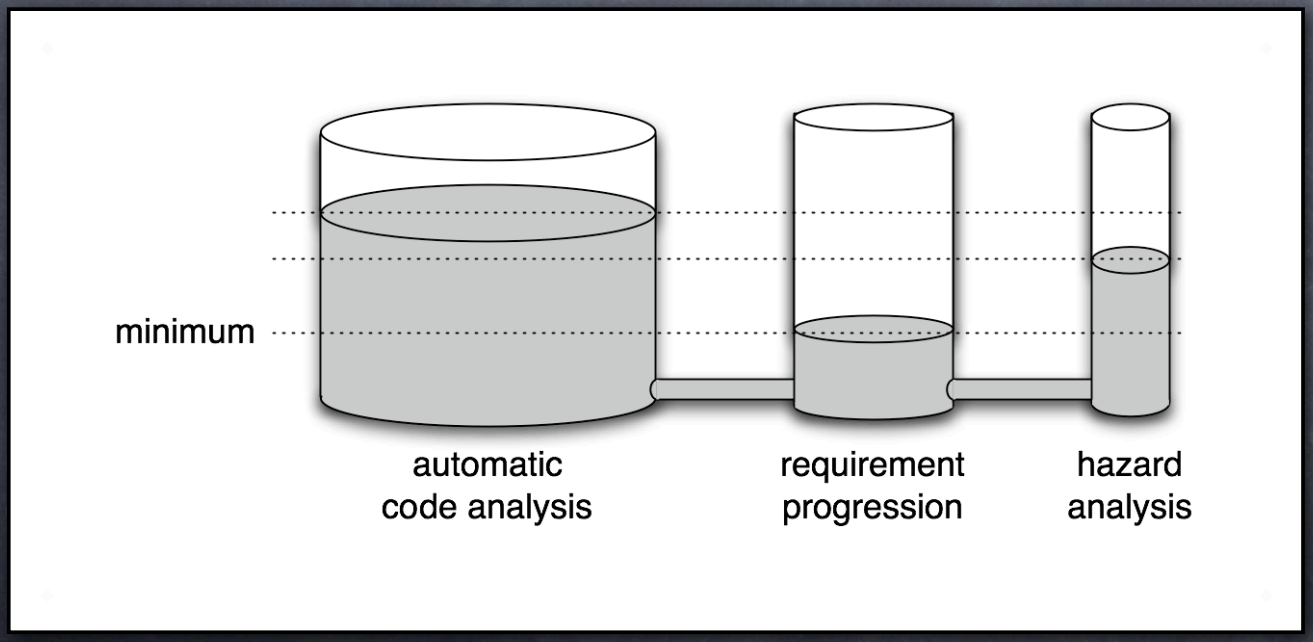


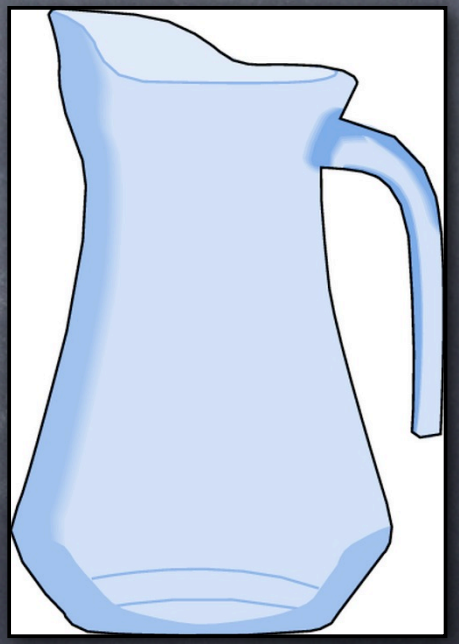
Idea: Water Glass Model



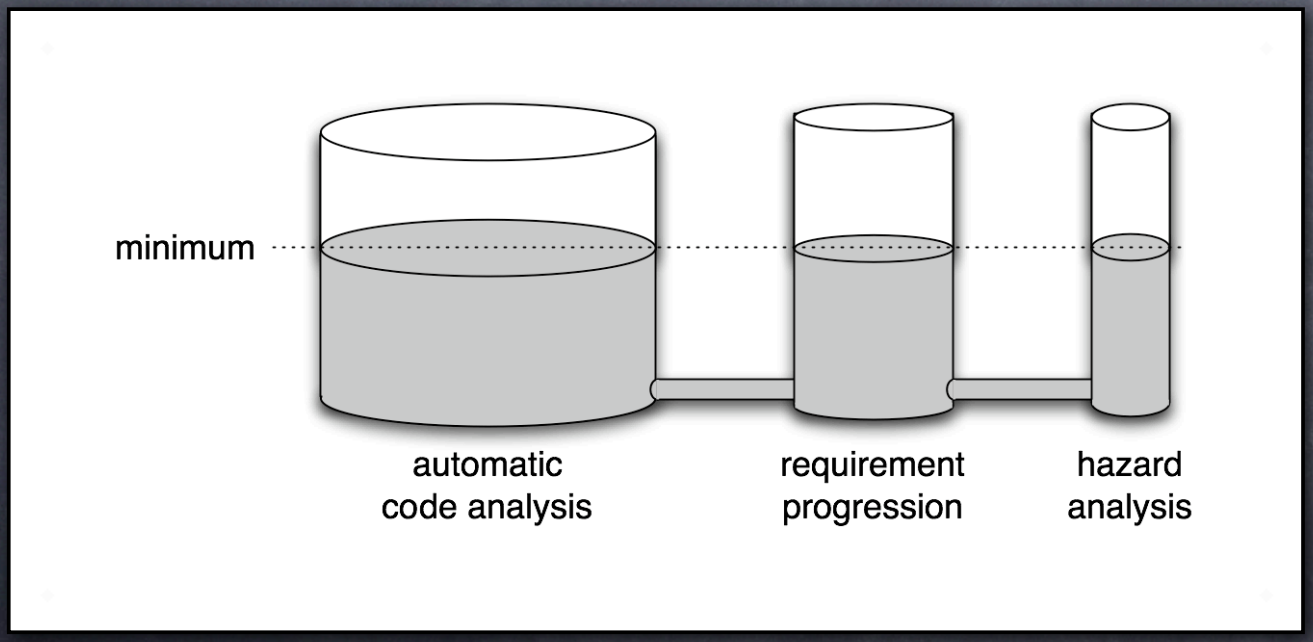


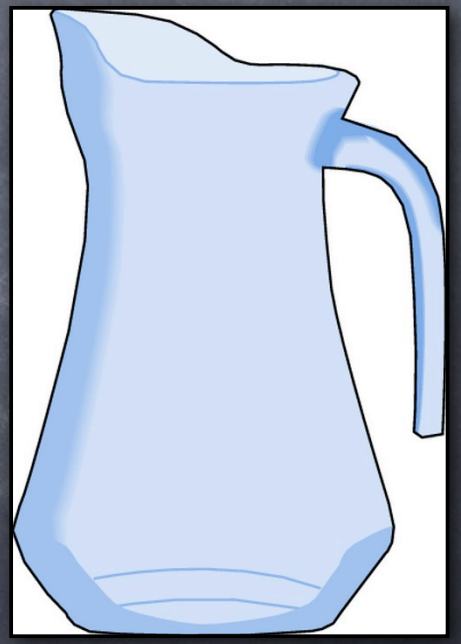
Idea: Water Glass Model



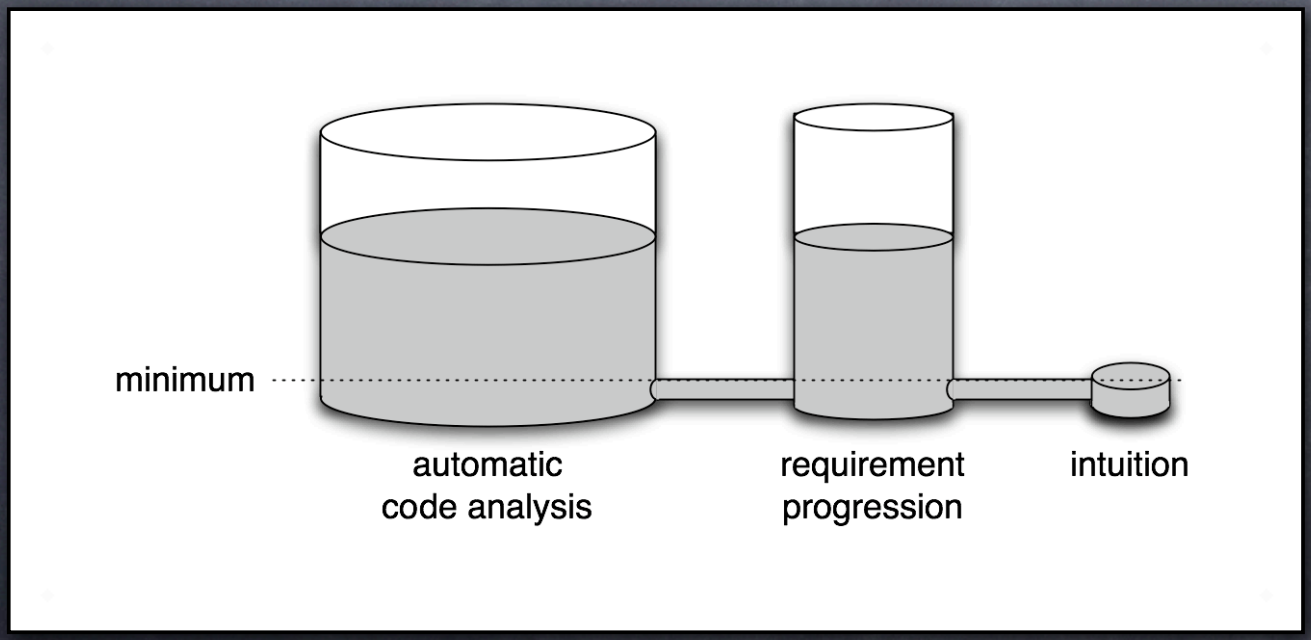


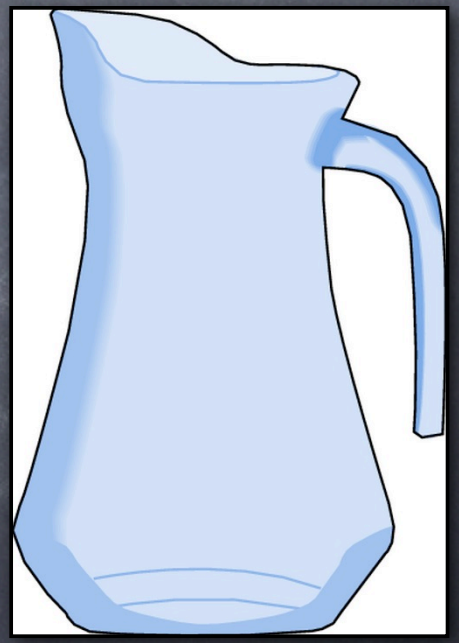
Idea: Water Glass Model



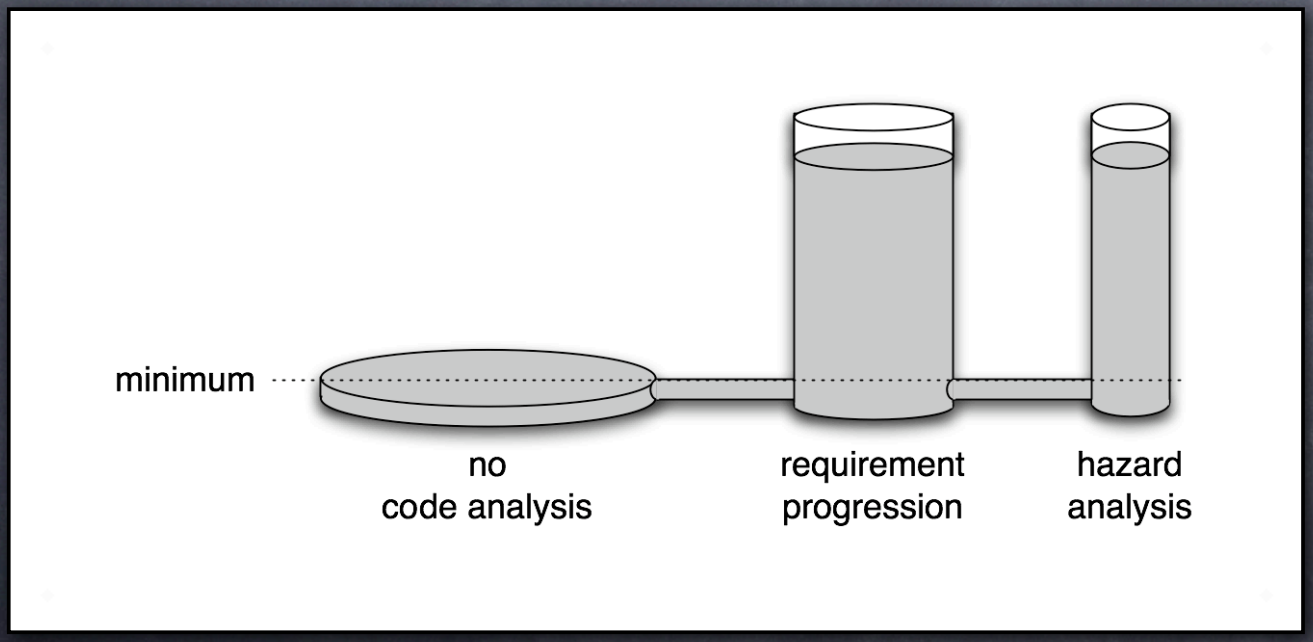


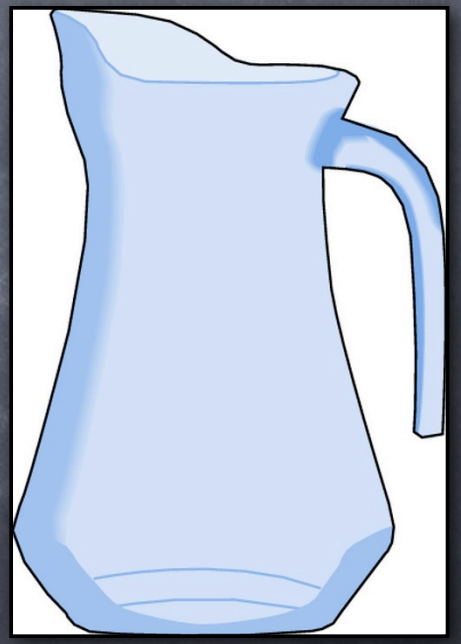
wrong requirements



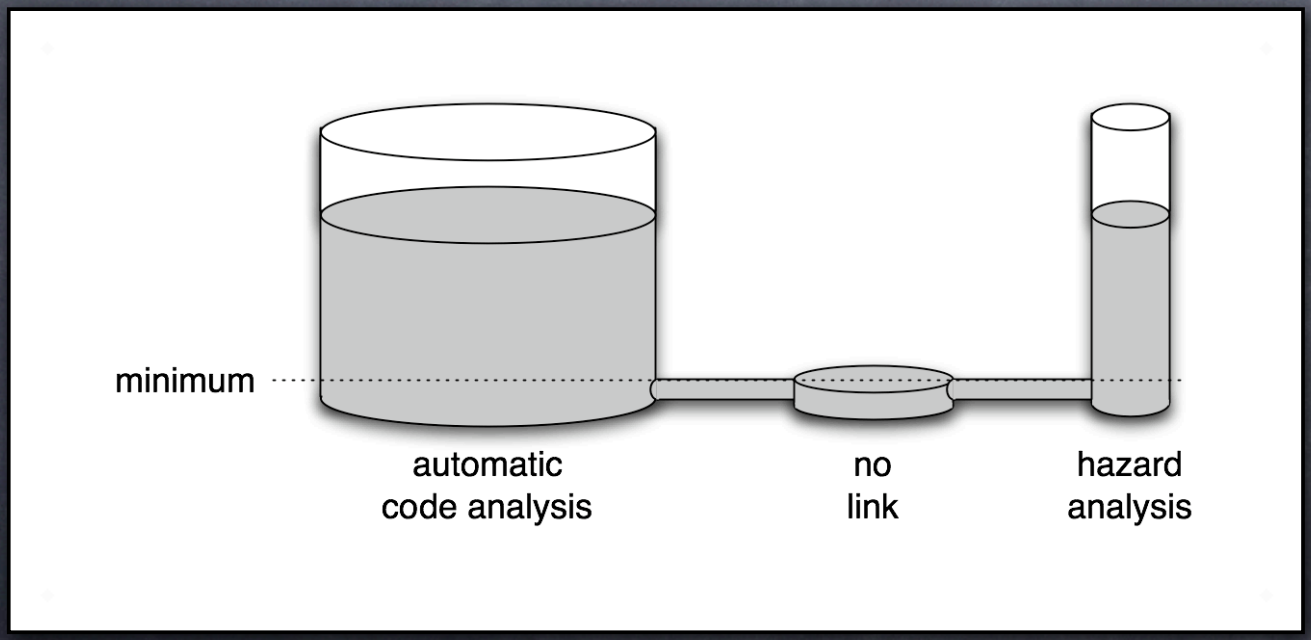


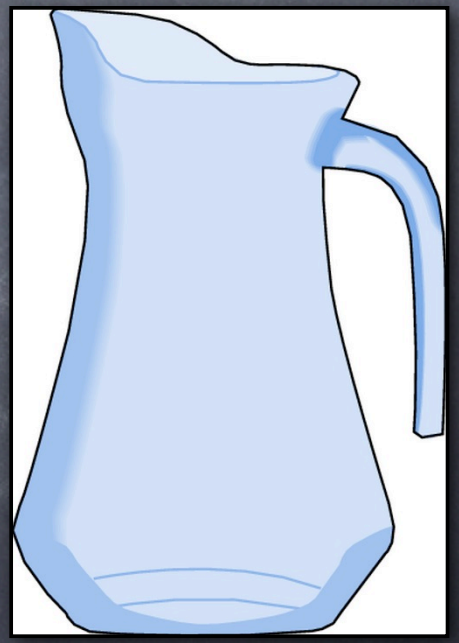
no substantiation



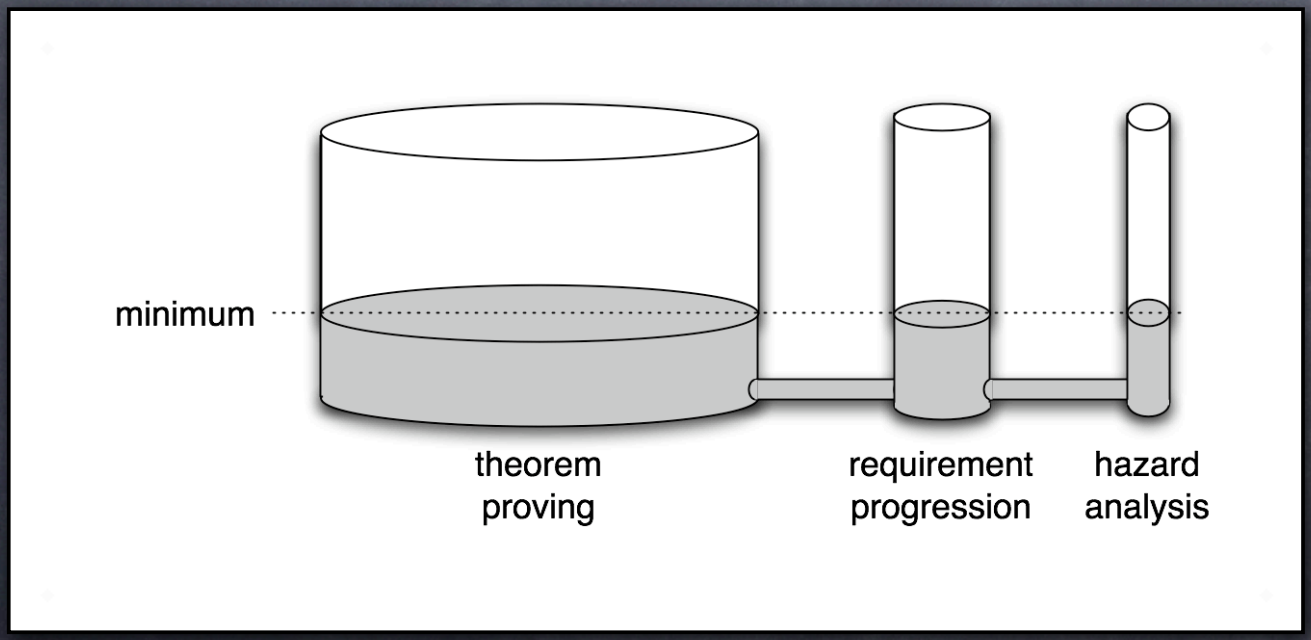


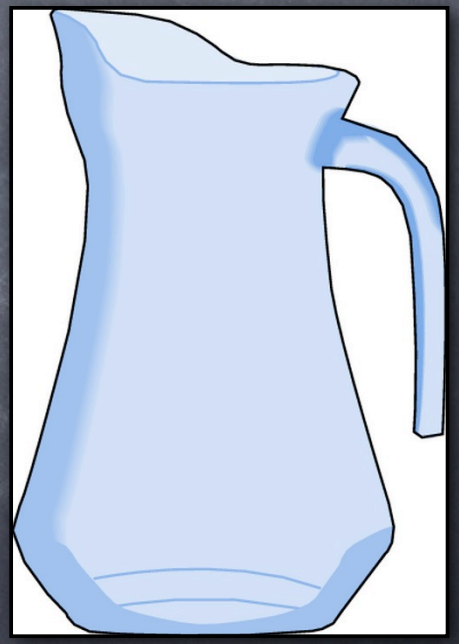
no link



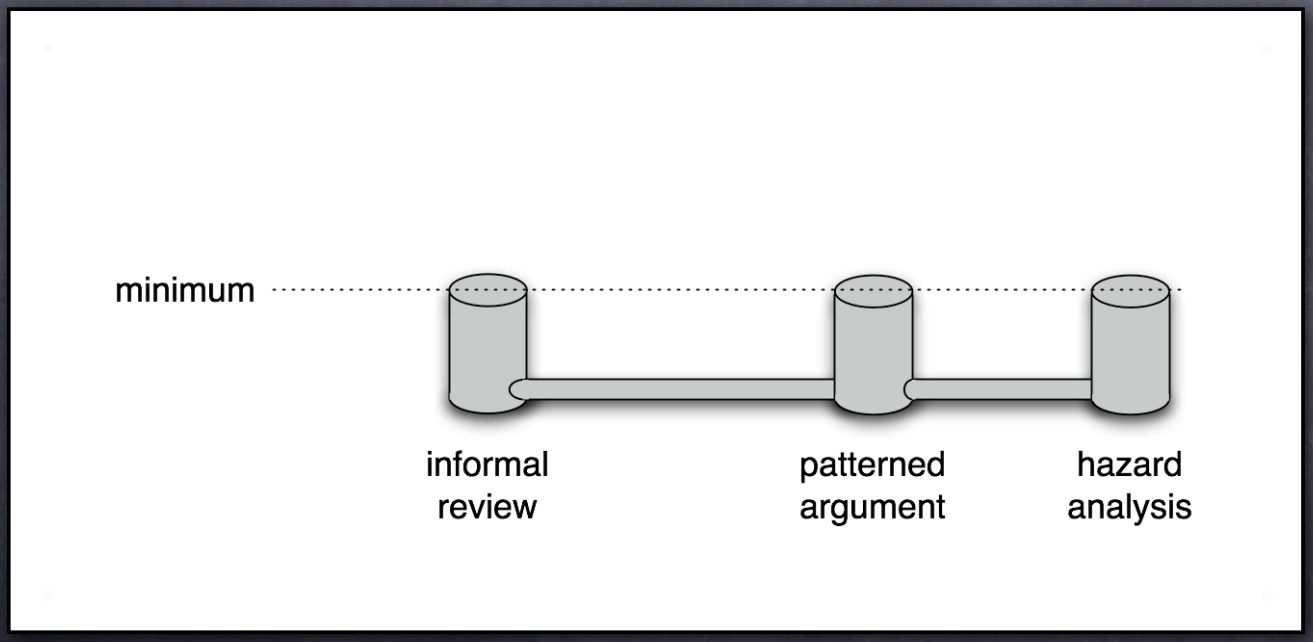


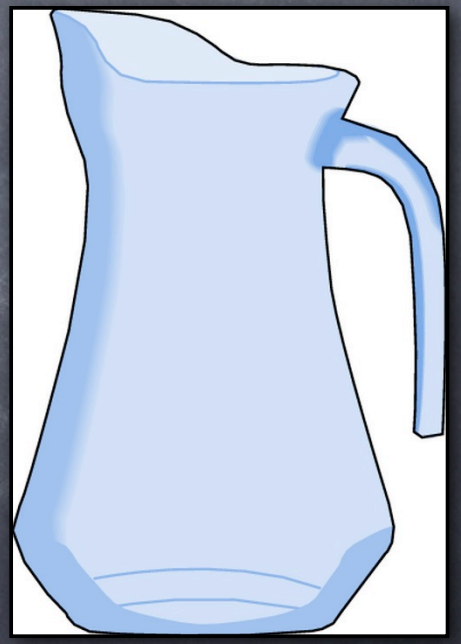
low budget, heavyweight techniques



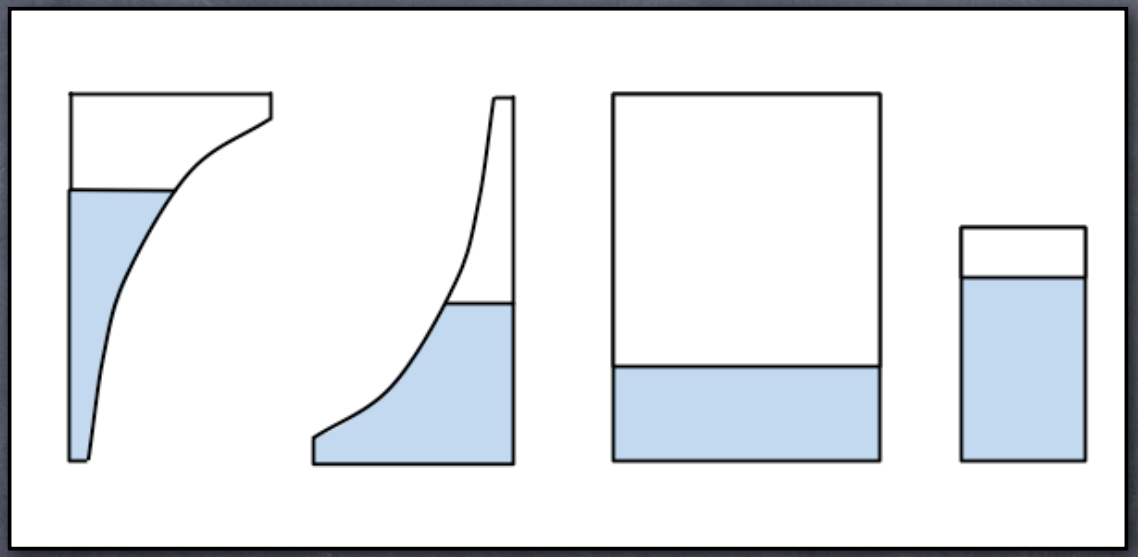


high budget, lightweight techniques





marginal returns



diminishing returns

high overhead

heavyweight

lightweight

Interests

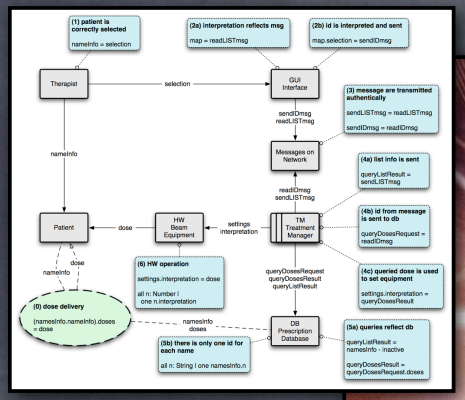
diagrams & techniques to interpret technical information and guide analysis

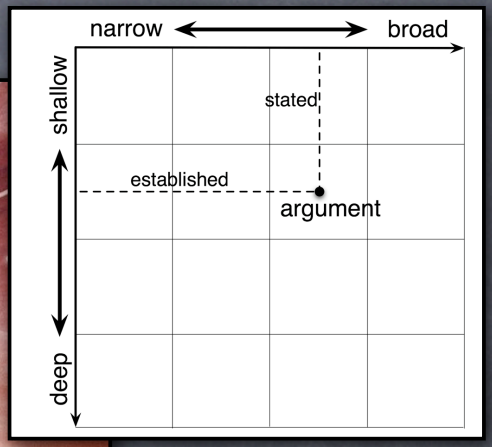
doctoral (requirements engineering)
- requirement progression guidance
- show secrecy attack traces

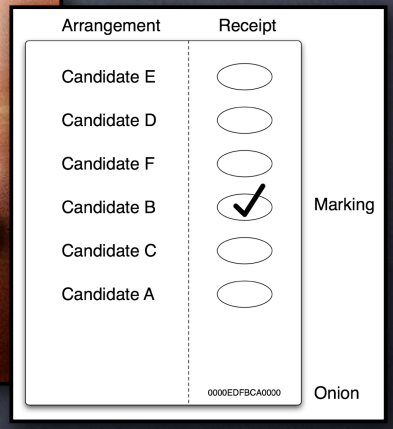
masters (modeling)
- role of constraint in model

side projects (information layout)
- magic layout for Alloy
- information representation in games

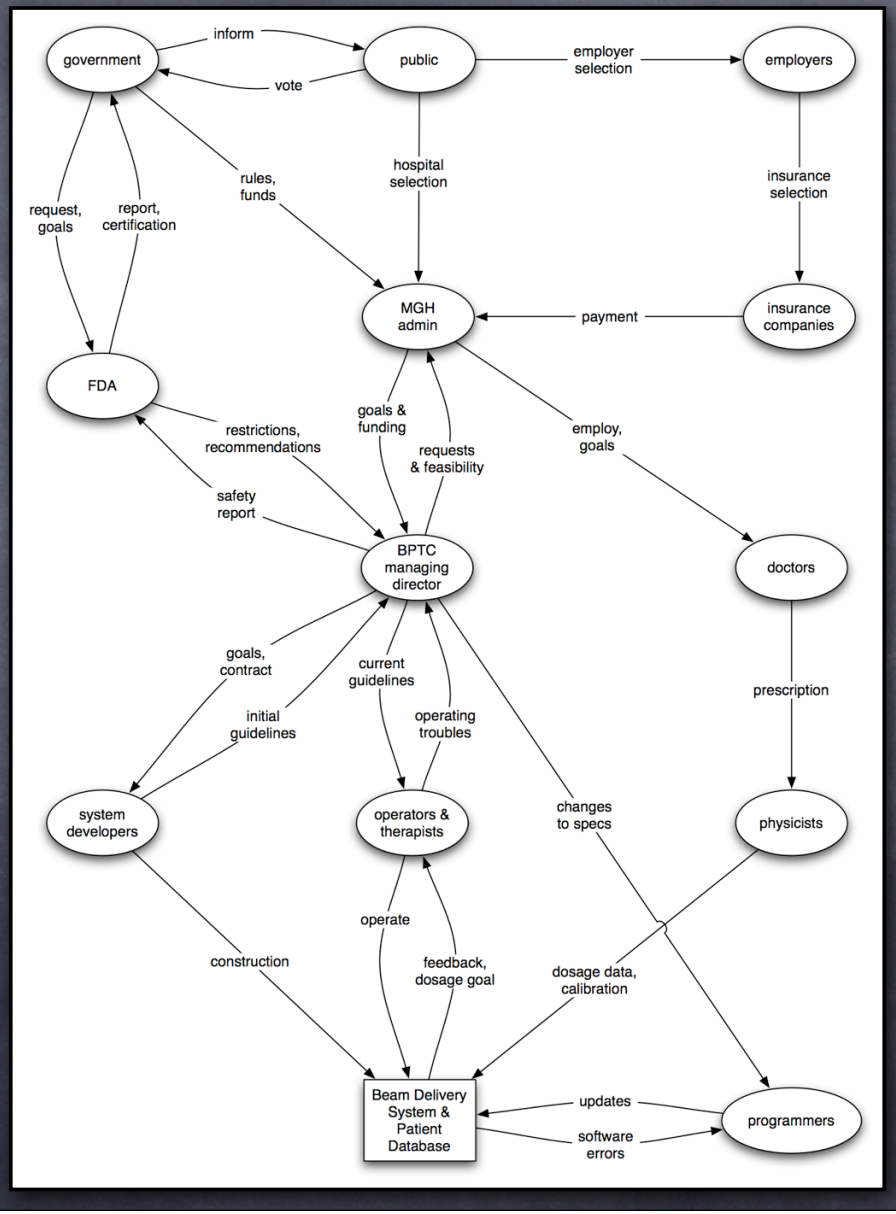
Questions?


(1) patient is correctly selected
nameInfo = selection
(2a) interpretation reflects msg
map = readLISTmsg
(2b) id is interpreted and sent
map.selection = sendIDmsg
Therapist
selection
GUI Interface
(3) message are transmitted authentically
sendLISTmsg = readLISTmsg
sendIDmsg = readIDmsg
sendIDmsg
readLISTmsg
nameInfo
Messages on Network
(4a) list info is sent
queryListResult = sendLISTmsg
readIDmsg
sendLISTmsg
Patient
dose
HW Beam Equipment
settings
interpretation
TM Treatment Manager
(4b) id from message is sent to db
queryDosesRequest = readIDmsg
nameInfo
dose
(6) HW operation
settings.interpretation = dose
all n: Number | one n.interpretation
queryDosesRequest
queryDosesResult
queryListResult
(4c) queried dose is used to set equipment
settings.interpretation = queryDosesResult
(0) dose delivery
(namesInfo.nameInfo).doses = dose
namesInfo
doses
DB Prescription Database
(5a) queries reflect db
queryListResult = namesInfo - inactive
queryDosesResult = queryDosesRequest.doses
(5b) there is only one id for each name
all n: String | one namesInfo.n


narrow
broad
shallow
deep
stated
established
argument


Arrangement
Receipt
Candidate E
Candidate D
Candidate F
Candidate B
Candidate C
Candidate A
Marking
0000EDFBCA0000
Onion

BPTC STAMP


government
inform
vote
public
employer selection
employers
request, goals
report, certification
rules, funds
hospital selection
insurance selection
MGH admin
payment
insurance companies
FDA
goals & funding
requests & feasibility
employ, goals
restrictions, recommendations
safety report
BPTC managing director
doctors
goals, contract
initial guidelines
current guidelines
operating troubles
prescription
system developers
operators & therapists
changes to specs
physicists
operate
construction
feedback, dosage goal
dosage data, calibration
Beam Delivery System & Patient Database
updates
software errors
programmers

Building PFs