

Dependability Arguments with Trusted Bases

Eunsuk Kang and Daniel Jackson
Massachusetts Institute of Technology



MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Radiation Therapy

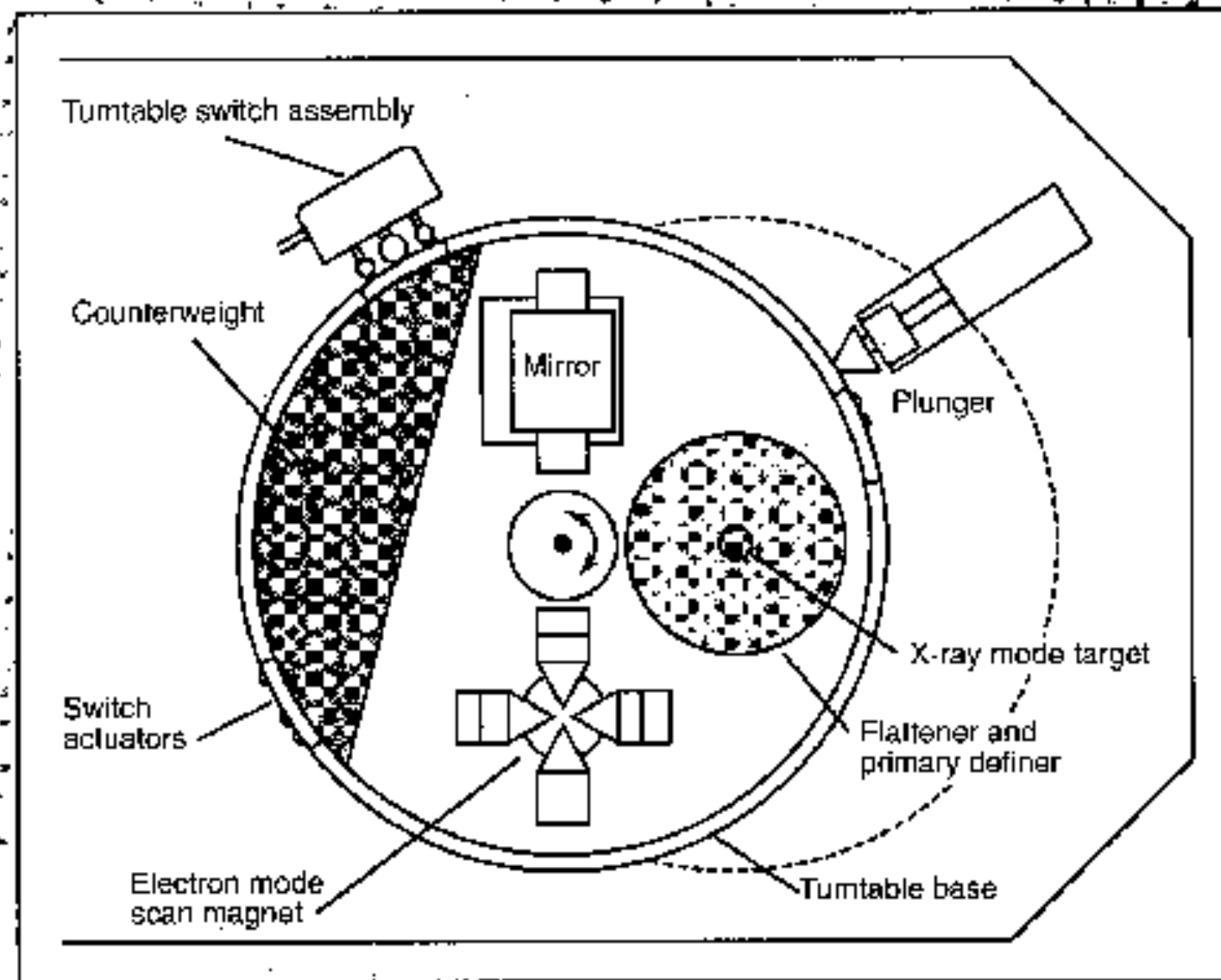


Figure B. Upper turntable assembly.

An Investigation of the Therac-25 Accidents
Nancy Leveson and Clark Turner (1993)

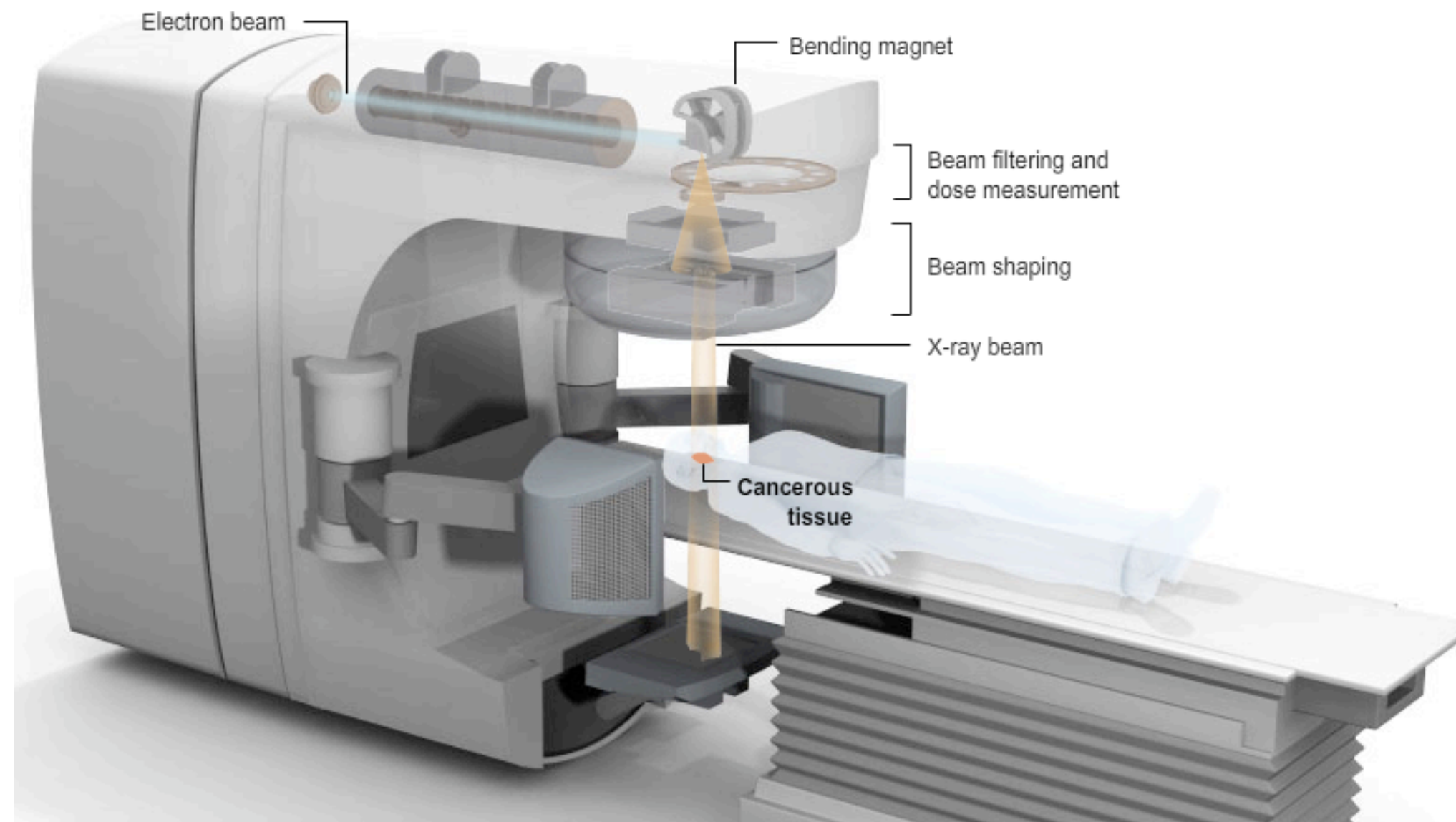
Lesson Learned?

THE RADIATION BOOM

Radiation Offers New Cures, and Ways to Do Harm

By WALT BOGDANICH

Published: January 23, 2010



“...[NY state] most stringent regulator of radioactive medical devices in the nation...”

“...**621** mistakes from January 2001 to January 2009...”

Achieving Dependability

Conventional

Process & standard

Post-facto analysis

“All or nothing”

Software for Dependable Systems: Sufficient Evidence?

National Academies Study

D. Jackson, M. Thomas, L. I. Millett (2007)

Achieving Dependability

Conventional

Process & standard

Post-facto analysis

“All or nothing”

Missing **direct** link?

Achieving Dependability

Conventional

Proposed

Process & standard

Explicit case

Post-facto analysis

“All or nothing”

Achieving Dependability

Conventional

Proposed

Process & standard

Explicit case

Post-facto analysis

Design for
dependability

“All or nothing”

Achieving Dependability

Conventional

Proposed

Process & standard

Explicit case

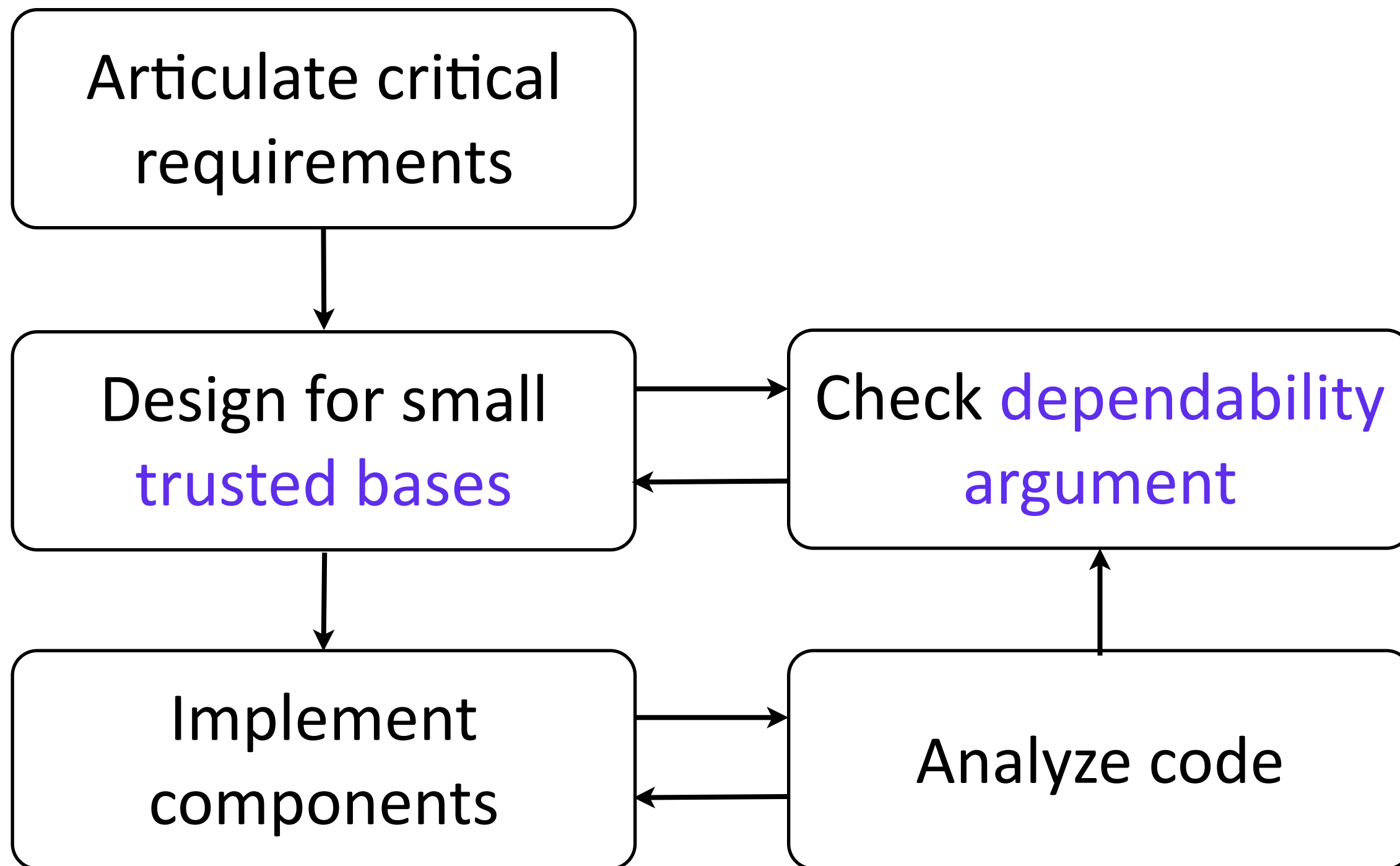
Post-facto analysis

Design for
dependability

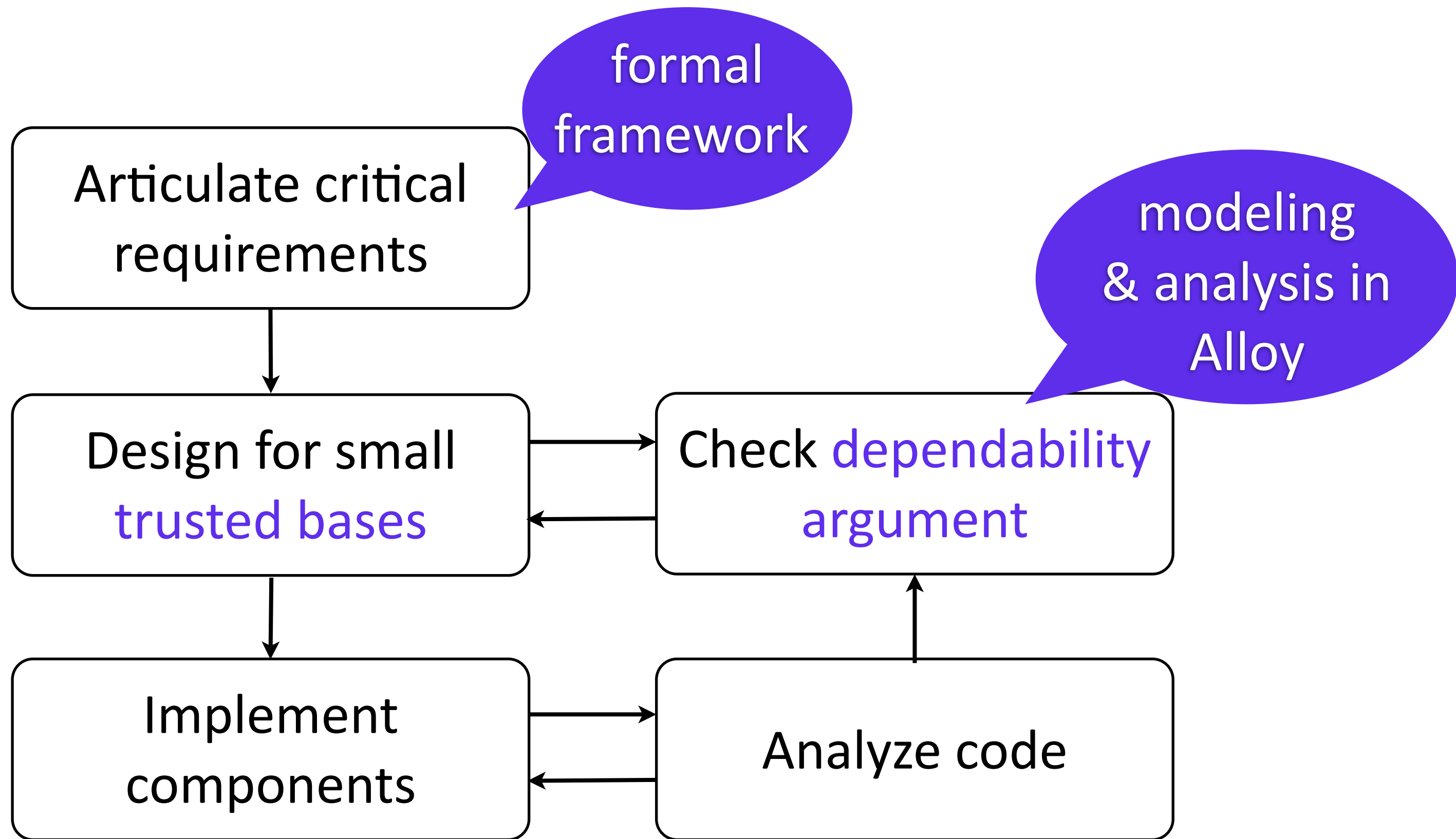
“All or nothing”

Most critical
requirements first

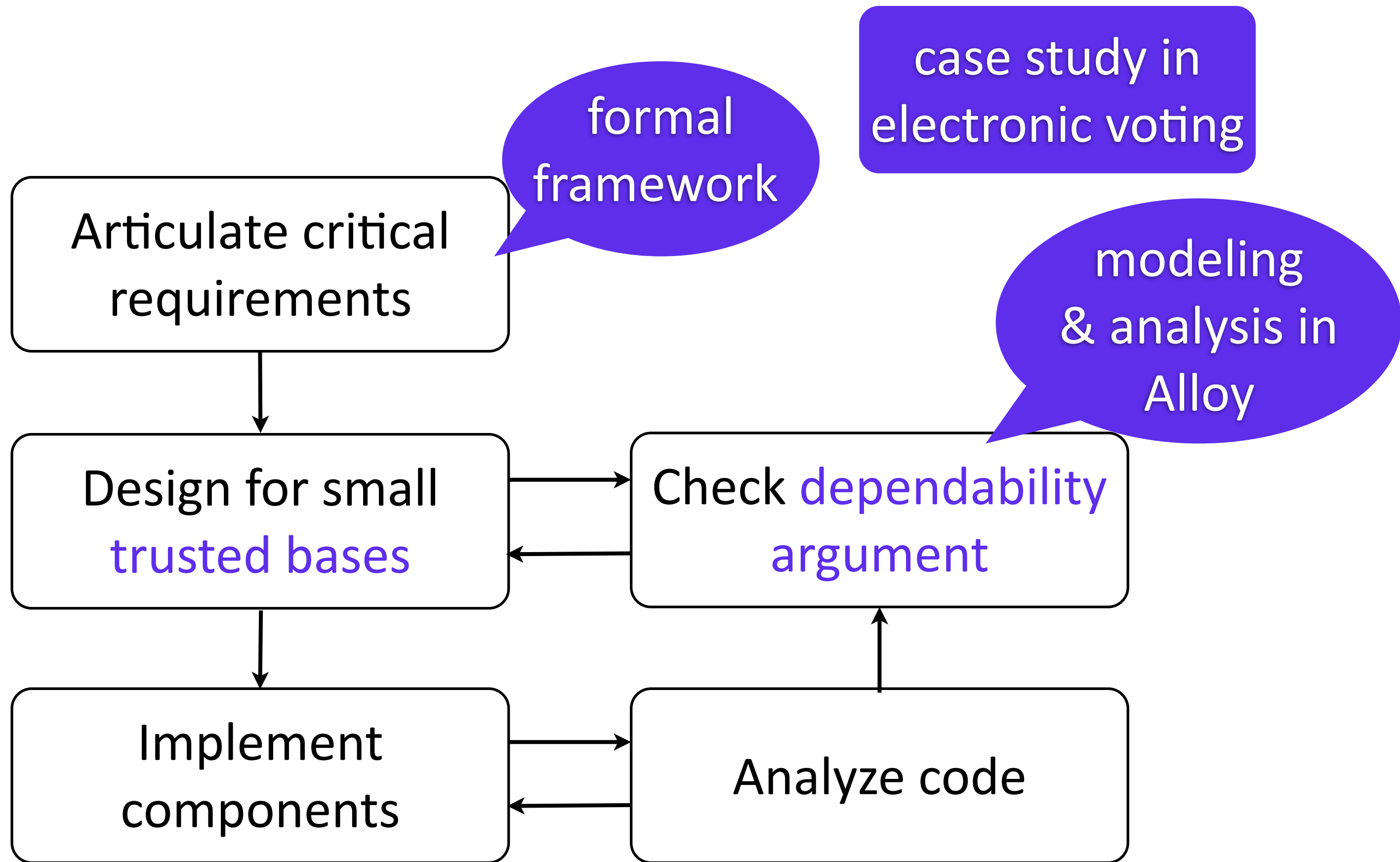
Methodology



In This Talk



In This Talk



Trusted Base

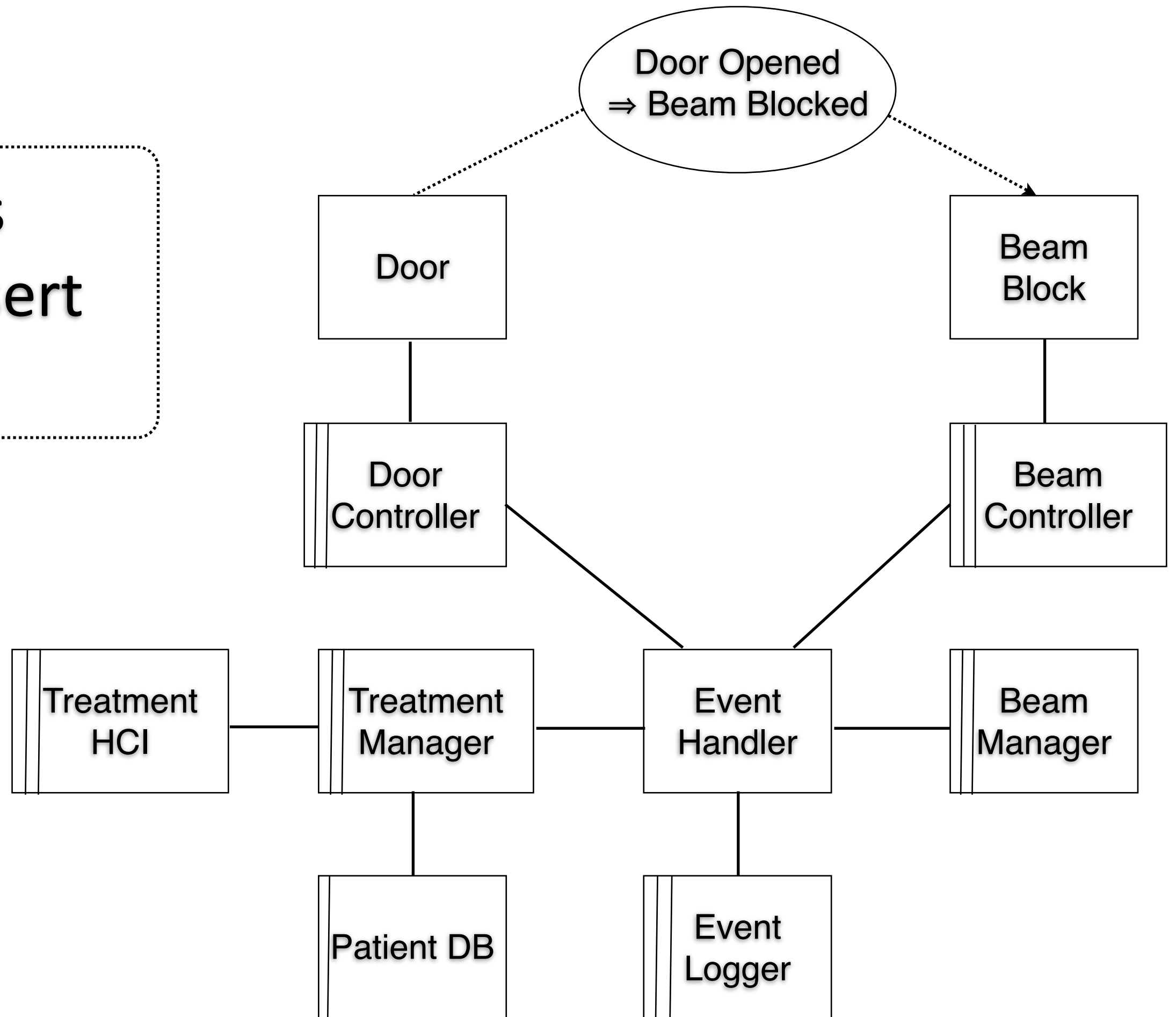
Example: Radiation Therapy



Francis H. Burr Proton Therapy Center
Massachusetts General Hospital

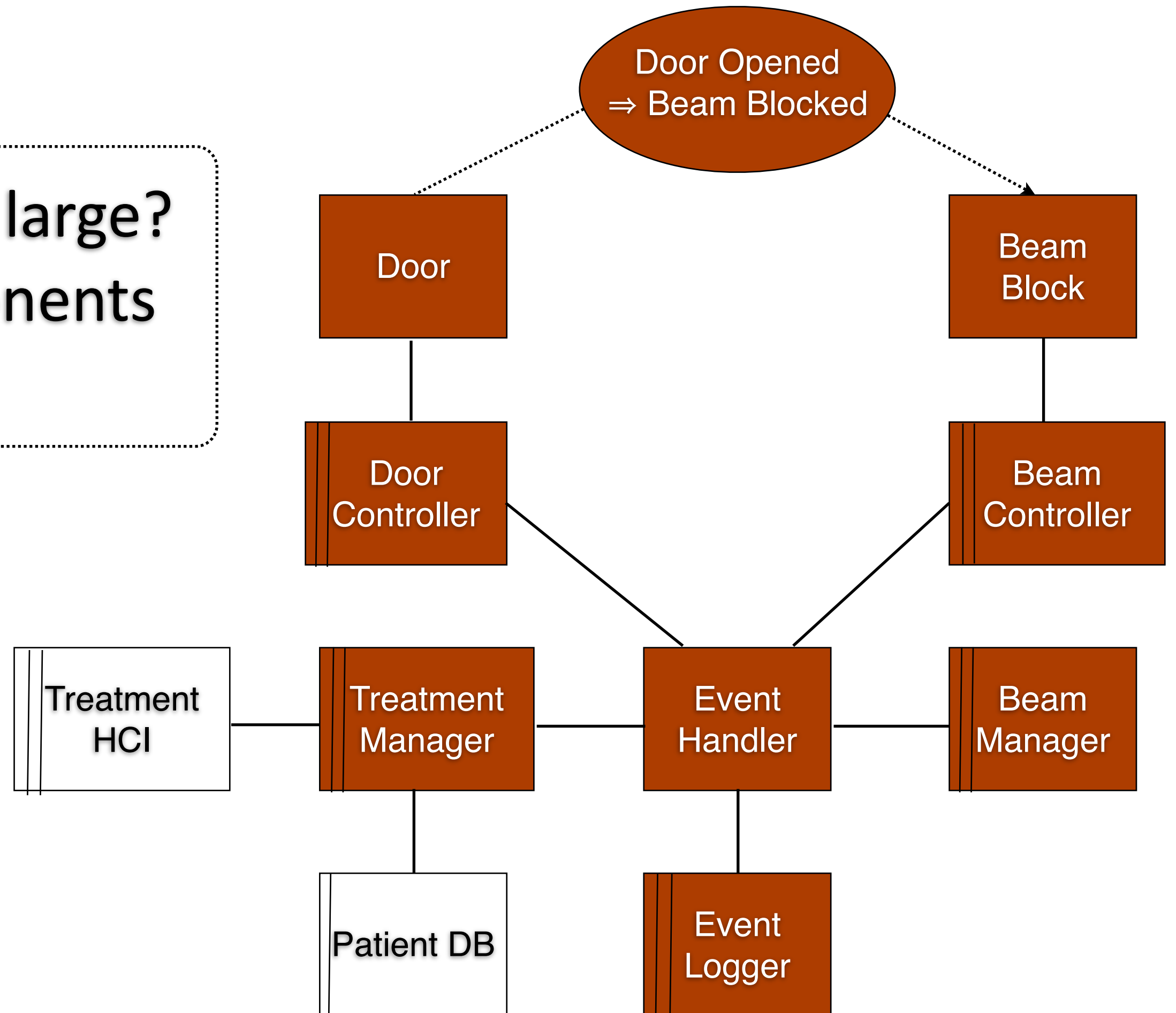
Door Safety

When the door is opened, must insert the beam block



Trusted Base

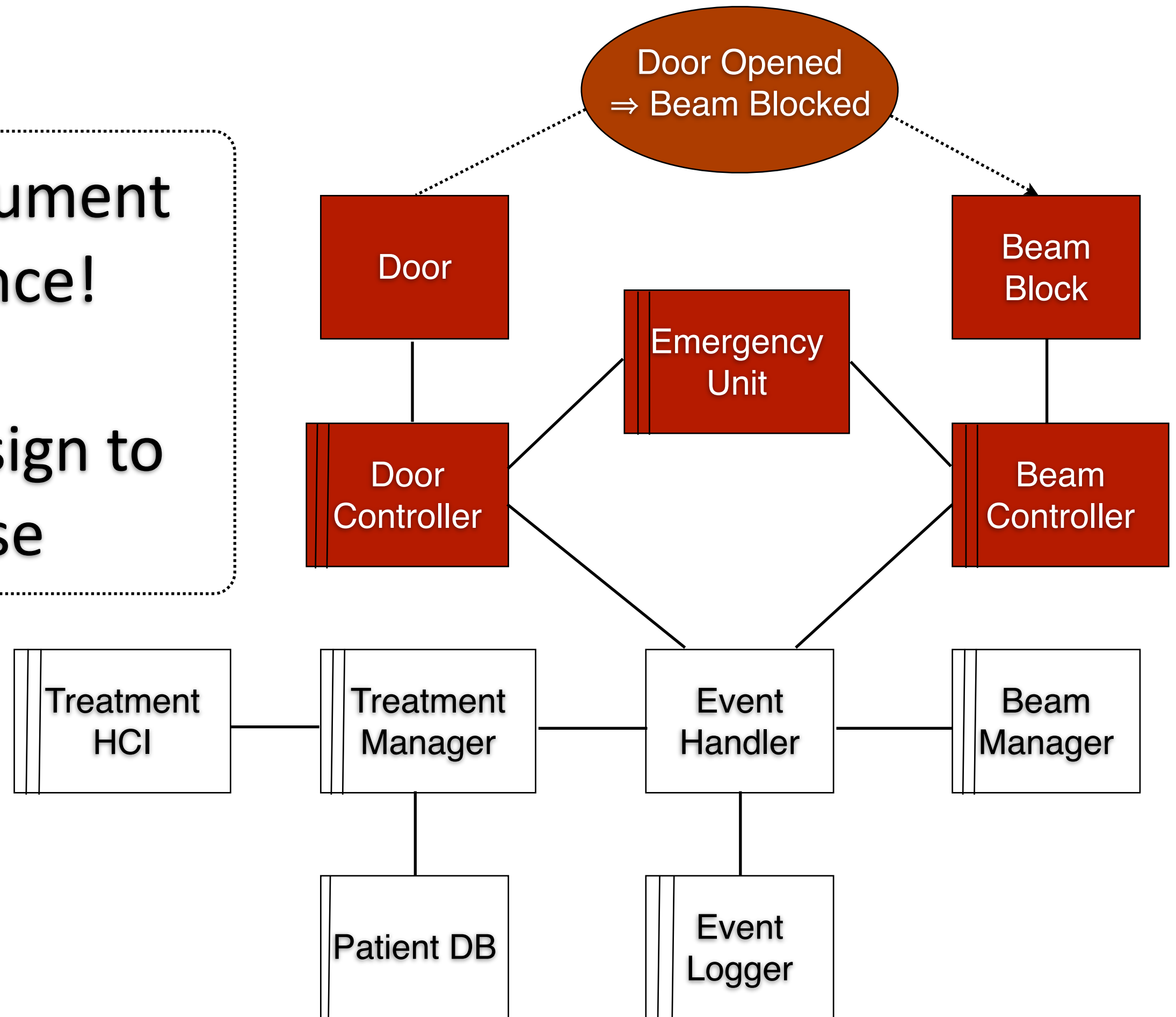
Trusted base too large?
Too many components
to check?



Alternative Design

Simpler safety argument
& greater confidence!

Key activity: Redesign to
reduce trusted base



Formal Framework

Formal notion

Each system consists of:

- domains D & machines M
- domain assumptions A & machine specifications S
- requirements R

Formal notion

Each system consists of:

- domains D & machines M
- domain assumptions A & machine specifications S
- requirements R

Mapping from each requirement to its trusted base

$$tb : R \rightarrow \wp(D \cup M)$$

Formal notion

Each system consists of:

- domains D & machines M
- domain assumptions A & machine specifications S
- requirements R

Mapping from each requirement to its trusted base

$$tb : R \rightarrow \wp(D \cup M)$$

Dependability argument for R_i

$$A_{tb(R_i)} \wedge S_{tb(R_i)} \Rightarrow R_i$$

$$\text{where } A_{tb(R_i)} = \bigwedge_{d \in tb(R_i)} (A_d) \text{ and } S_{tb(R_i)} = \bigwedge_{m \in tb(R_i)} (S_m)$$

Requirements Satisfaction

Traditional notion

$$A \wedge S \Rightarrow R$$

Proposed notion

$$\forall R_i \in R \cdot (A_{tb(R_i)} \wedge S_{tb(R_i)} \Rightarrow R_i)$$

Requirements Satisfaction

Traditional notion

$$A \wedge S \Rightarrow R$$

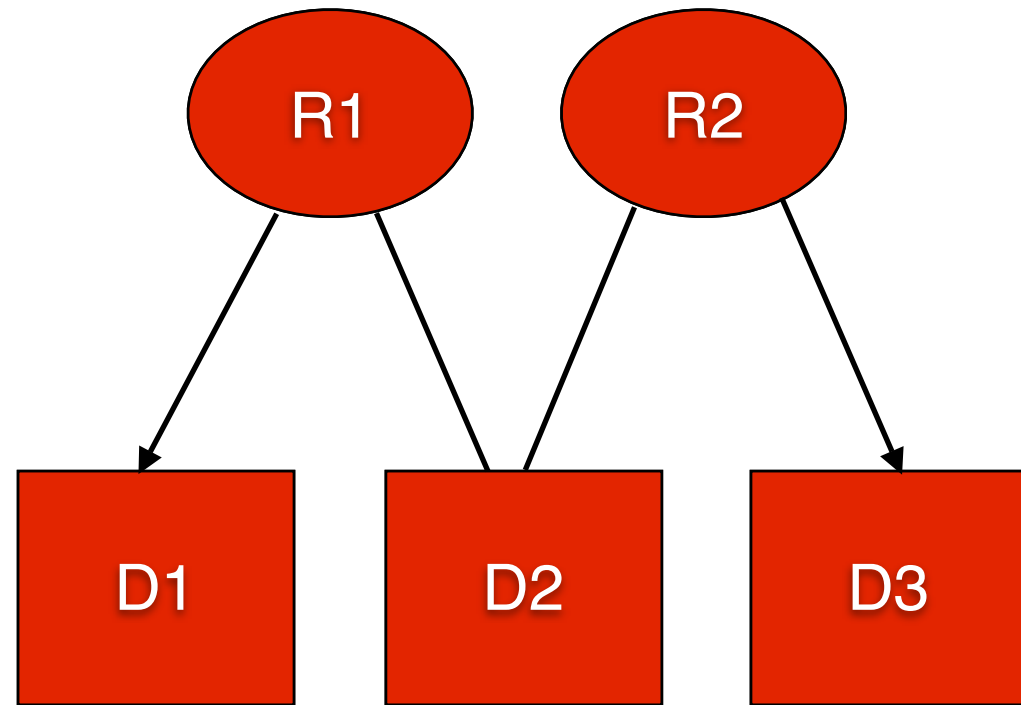
Proposed notion

$$\forall R_i \in R \cdot (A_{tb(R_i)} \wedge S_{tb(R_i)} \Rightarrow R_i)$$

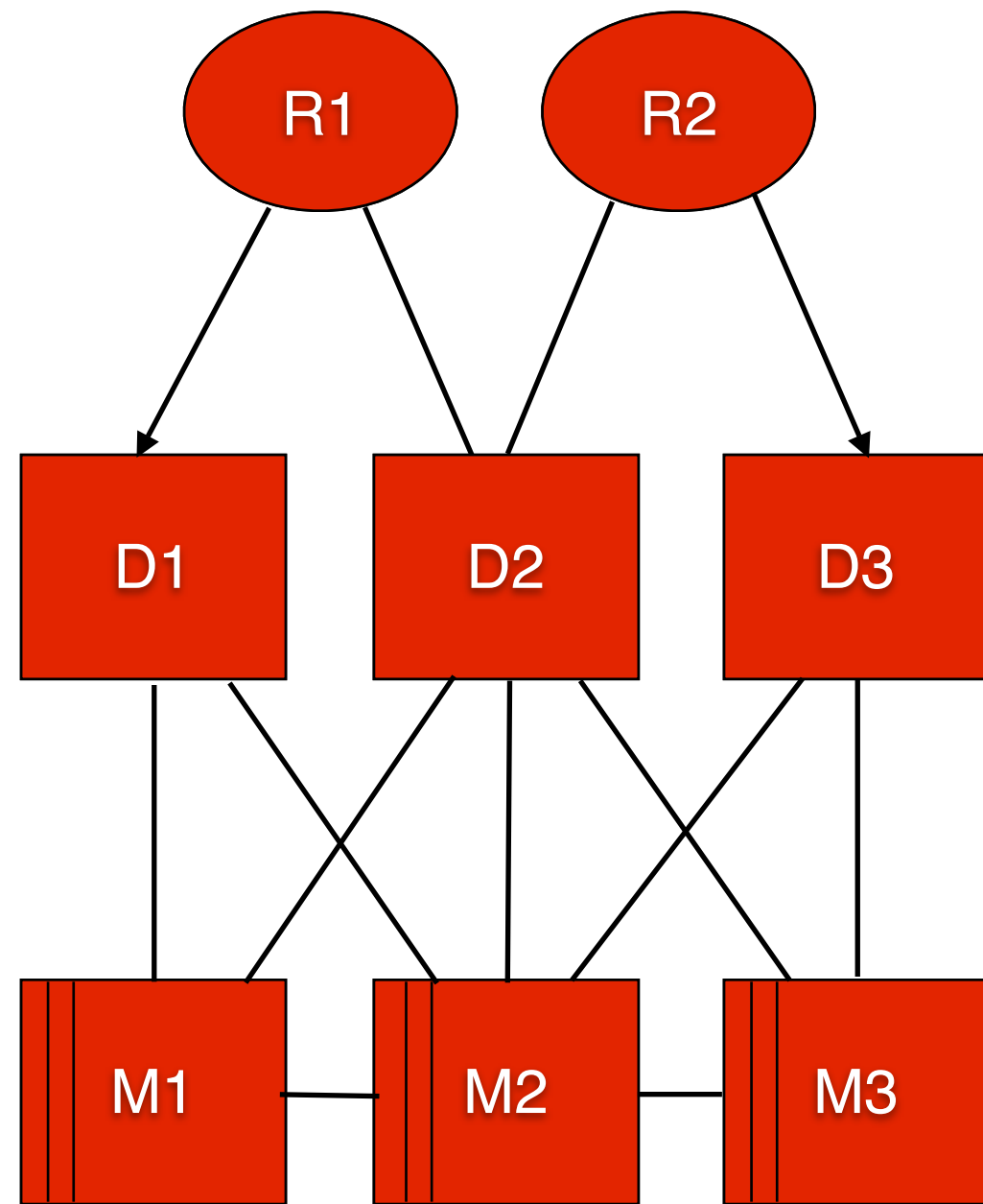
Traditional notion often too hard to achieve!

- unrealistic assumptions (e.g. no operator mistakes)
- some specs too difficult or expensive
- simply too many components!

Separation of Requirements

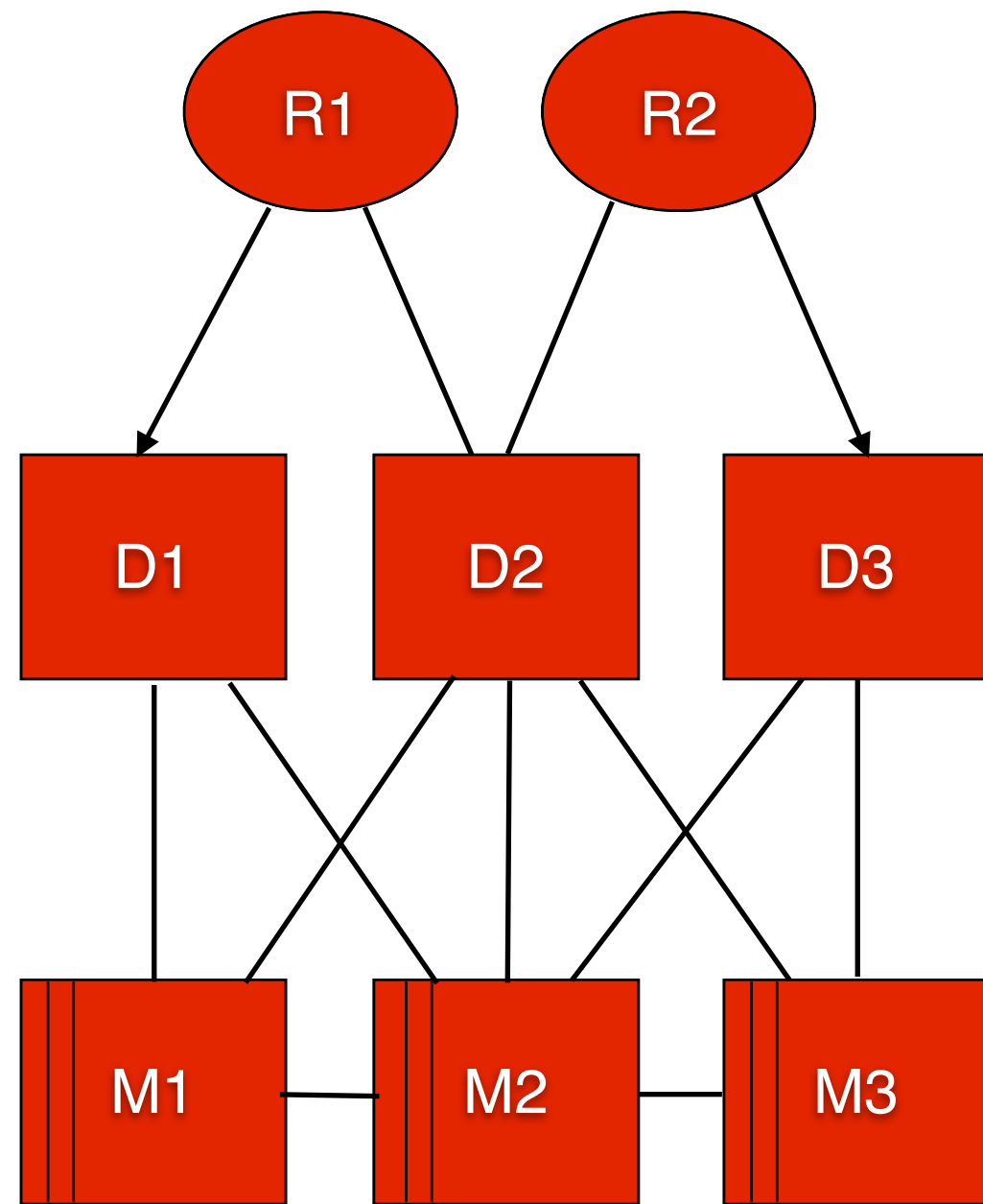


Separation of Requirements

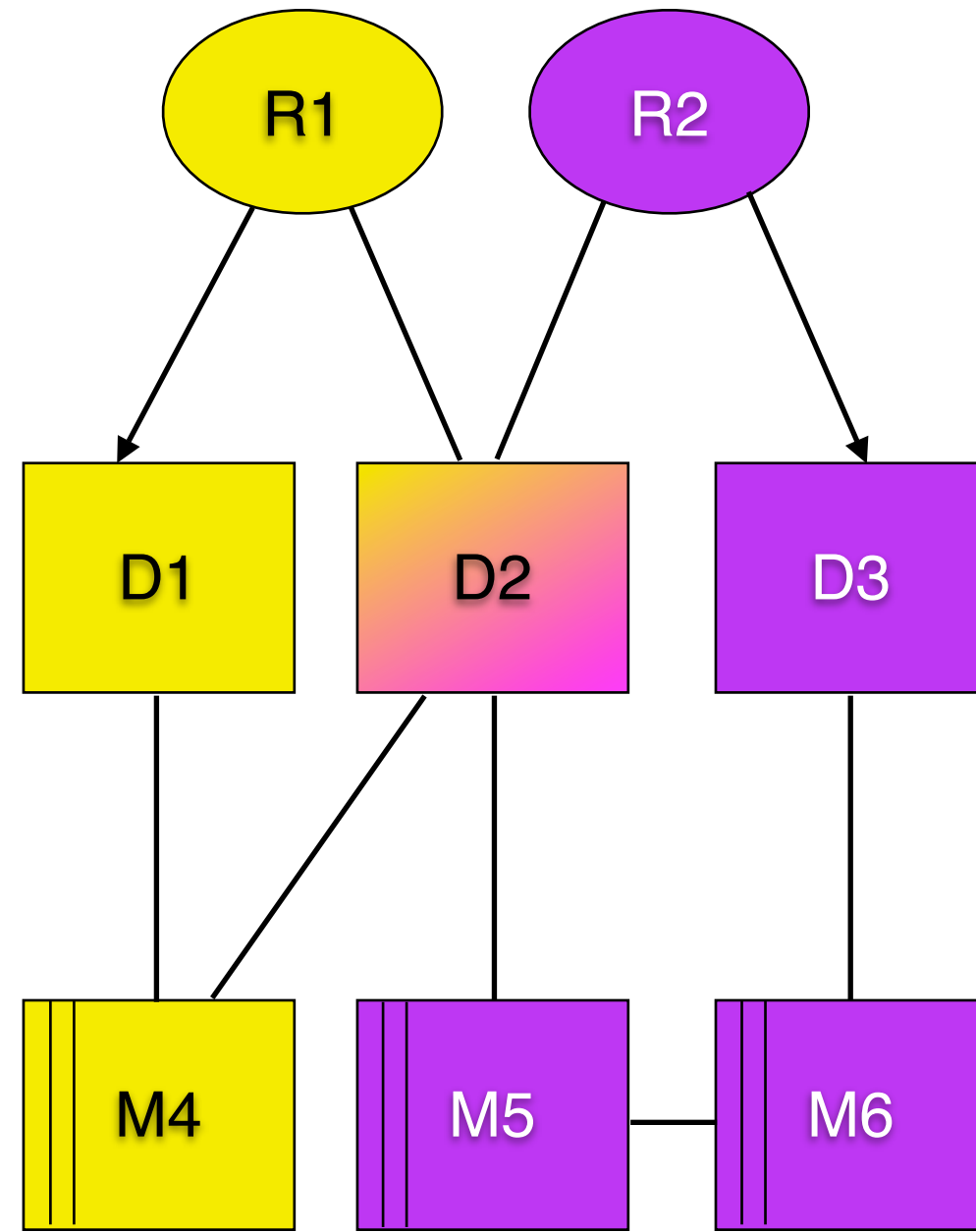


Design X

Separation of Requirements

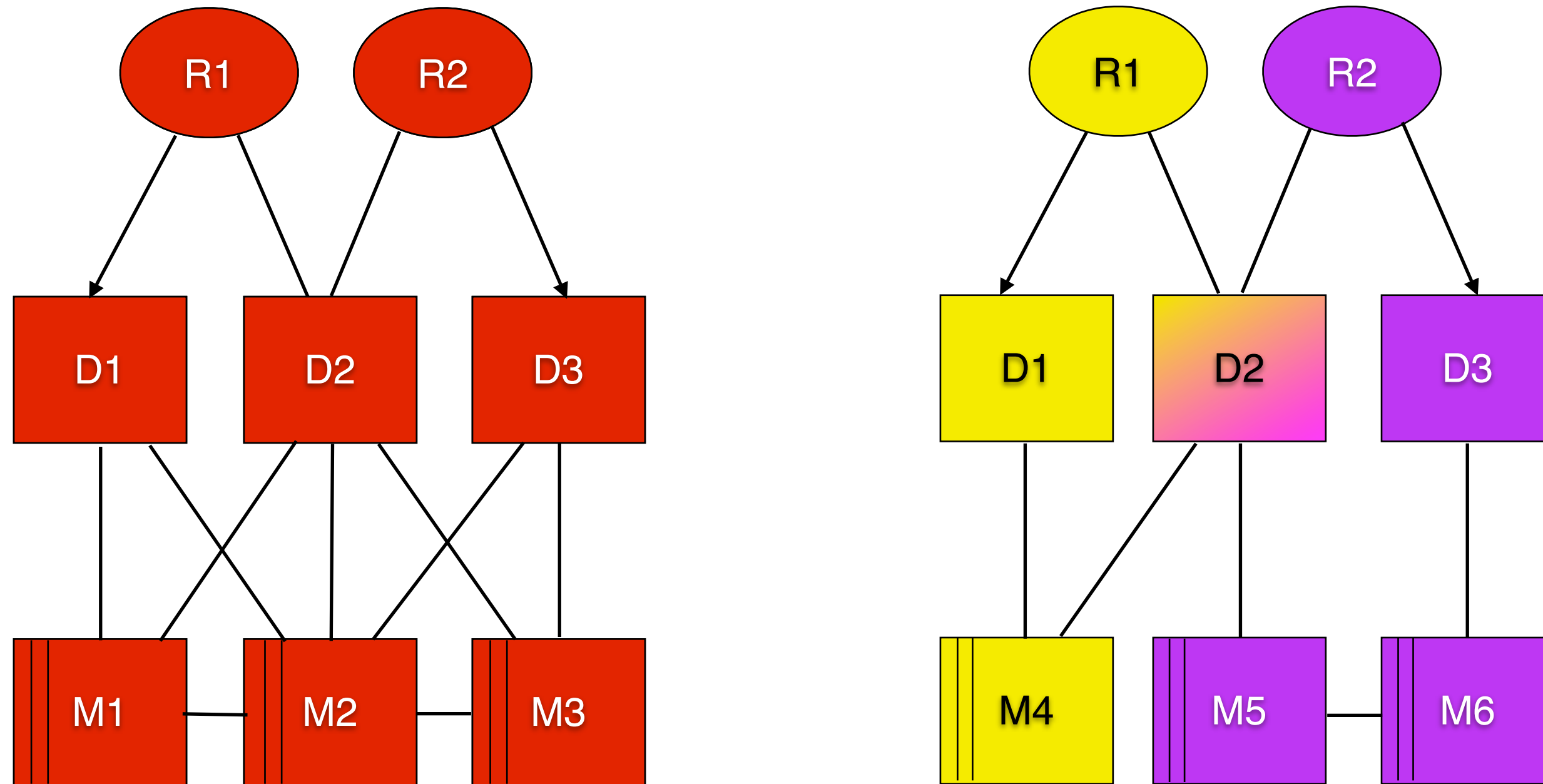


Design X



Design Y

Separation of Requirements



Decoupling in design cannot be achieved without decoupling in requirements!

Modeling & Analysis

Modeling & Analysis

Why model?

- articulate requirements, assumptions, specs
- check arguments & find mistakes

Alloy

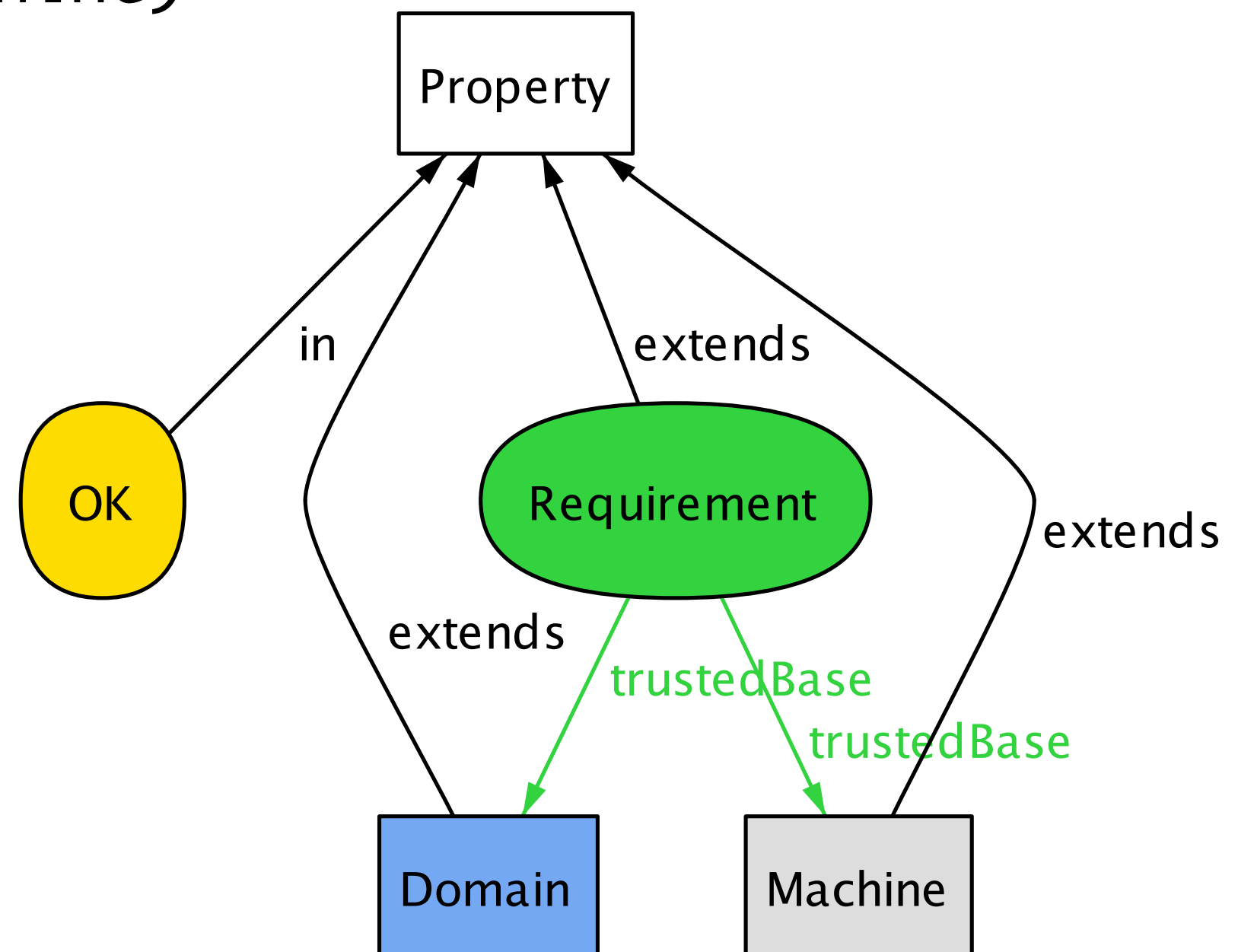
- flexibility for encoding idioms
- support for automated analysis
- other languages may also be suitable!

Framework in Alloy

```
abstract sig Property {}  
sig OK in Property {}
```

```
abstract sig Domain extends Property {}  
abstract sig Machine extends Property {}  
abstract sig Requirement extends Property {  
  trustedBase : set (Domain + Machine)  
}
```

```
assert DependabilityArgument {  
  all r : Requirement |  
    r.trustedBase in OK implies  
    r in OK  
}
```



Example: Reliable File Transfer

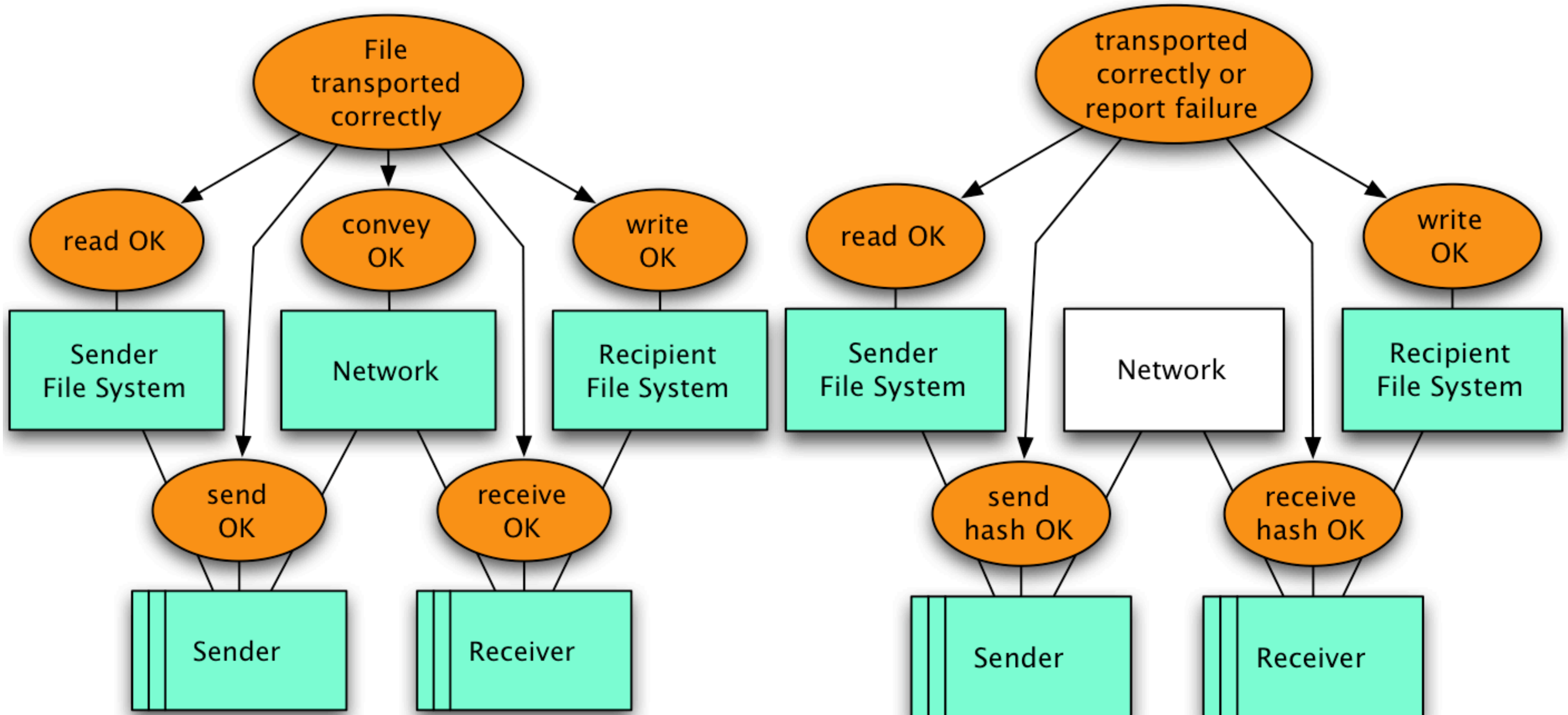
Requirement: Transfer a file from one host to another



End-to-End Principles in System Design

Jerome Saltzer, David Reed, and David Clark (1984)

Two FTP Designs



Standard Design

End-to-End Design

File Transfer in Alloy

```
abstract sig Packet {}
sig Block extends Packet {}
sig Hash extends Packet {}
sig File {
  blocks : set Block
  hash : Hash
}
```

Sender Host

```
sig SenderFileSys extends Domain {  
  file : File,  
  app : SenderApp  
}{  
  this in OK iff app.readFile = file  
}
```

Assign a property
to a component



```
sig Network extends Domain {  
  packetsIn, packetsOut : set Packet  
}{  
  this in OK iff packetsIn = packetsOut  
}
```

Hashing Property

```
fact HashingProperties {  
  all f1, f2 : File |  
    f1.hash = f2.hash iff f1.blocks = f2.blocks  
}
```

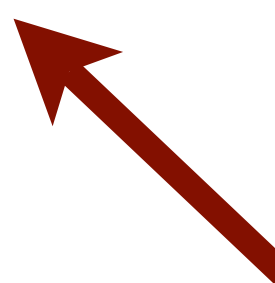


No two files have
the same hash

Requirement

```
sig ReliableTransferReq extends Requirement {
  senderFileSys : SenderFileSys,
  receiverFileSys : ReceiverFileSys,
  receiverApp : ReceiverApp
}{
  this in OK iff
    senderFileSys.file.blocks = receiverFileSys.file.blocks or
    receiverApp.receivedHash != receiverApp.computedHash

  trustedBase = {senderFileSys + senderFileSys.app +
                 receiverFileSys + receiverFileSys.app}
}
```



Assigning trusted base
to a requirement

Analysis

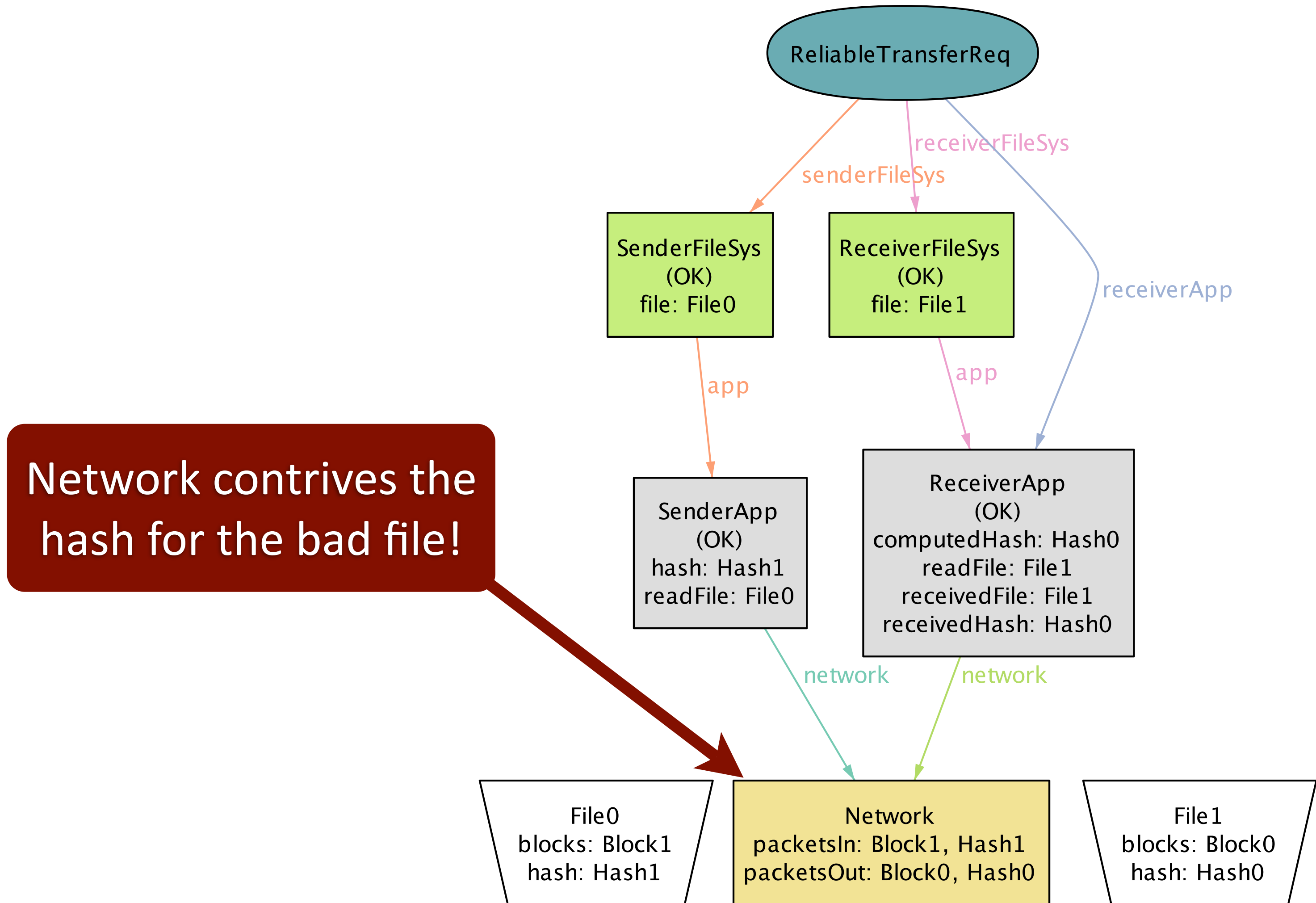
```
assert DependabilityArgument {  
  all r : Requirement |  
    r.trustedBase in OK implies r in OK  
}
```

```
check DependabilityArgument for 5
```



State the assertion to
be checked

Counterexample



Fixing the Model

```
fact HashingProperties {  
  all f1, f2 : File |  
    f1.hash = f2.hash iff f1.blocks = f2.blocks  
  
  all h : Hash & Network.packetsOut |  
    h in Network.packetsIn or no f : File | h = f.hash  
}
```



New property of
hashing added

Case Study

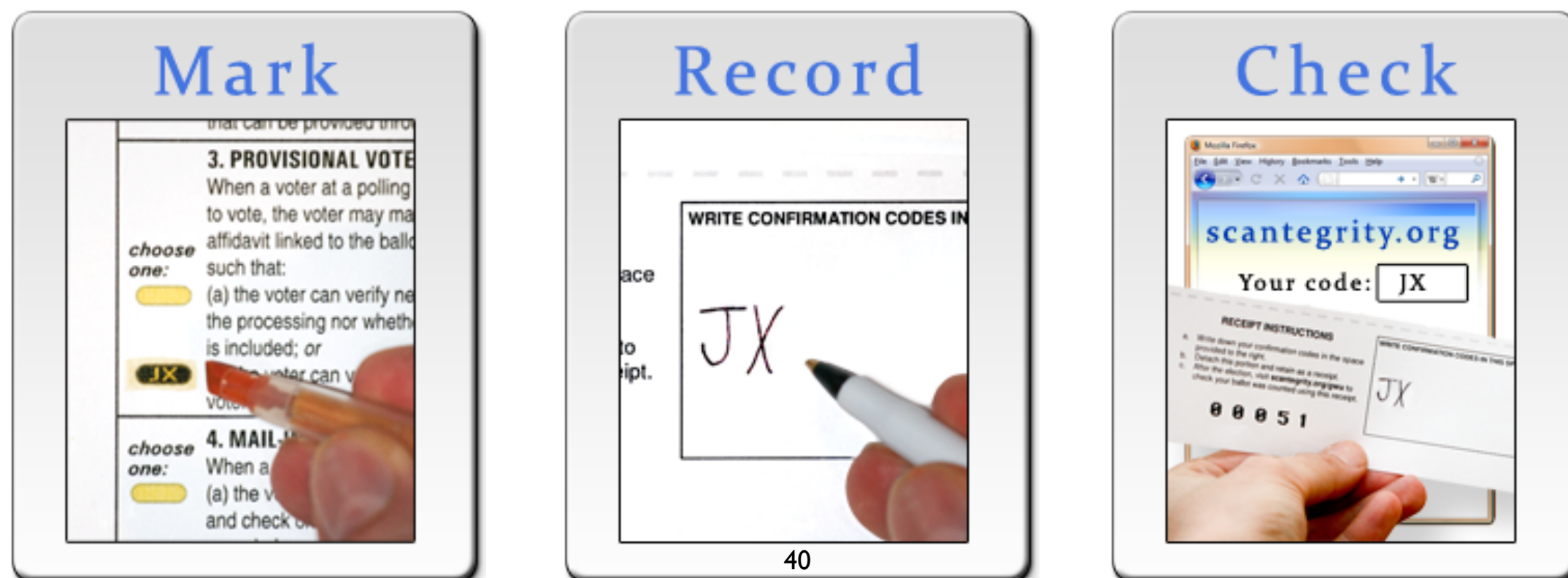
Electronic Voting

Two designs

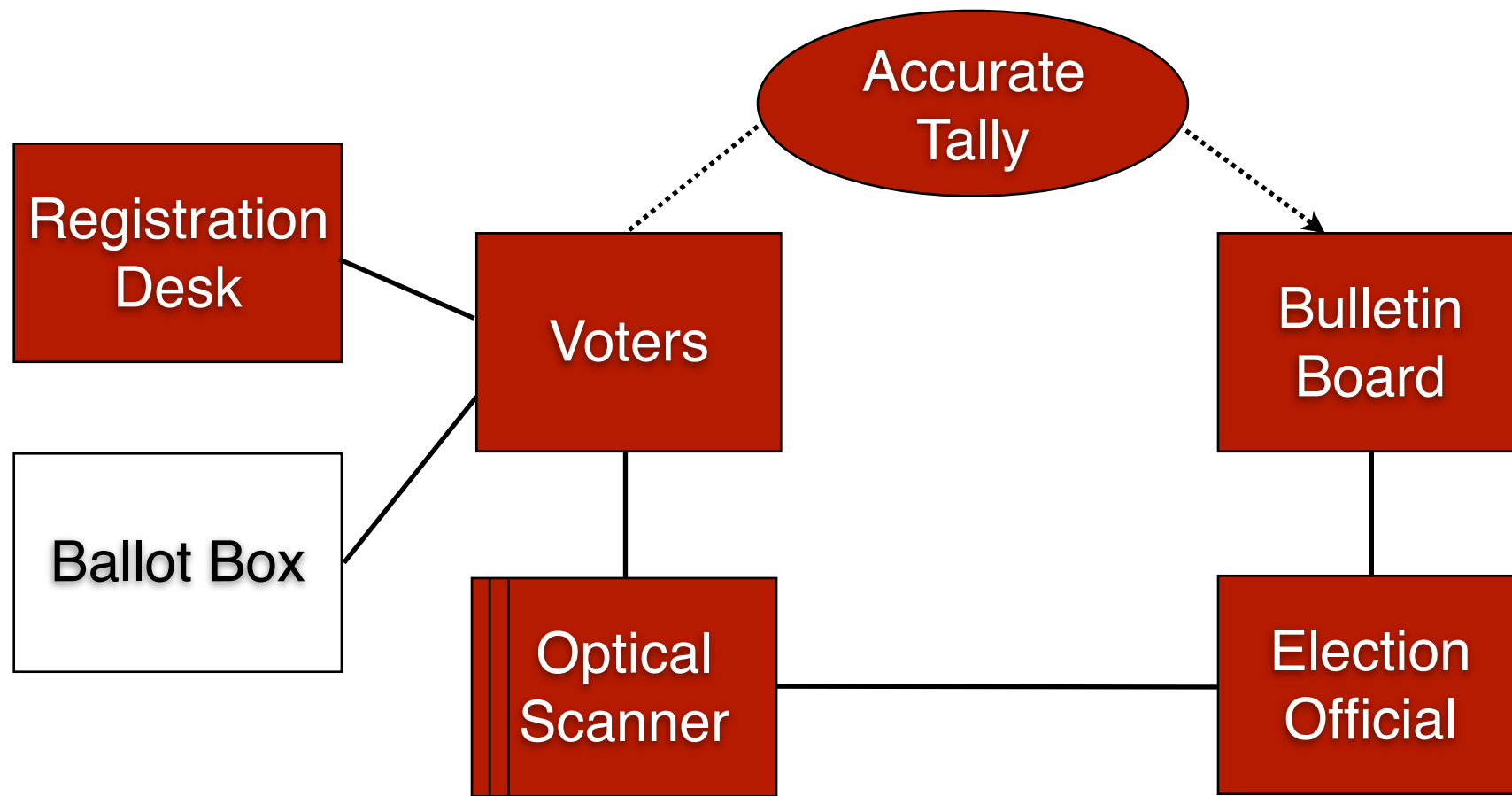
- **Optical scanner system**: shown vulnerabilities
- **Scantegrity**: *end-to-end* (E2E) verifiable system

Case study

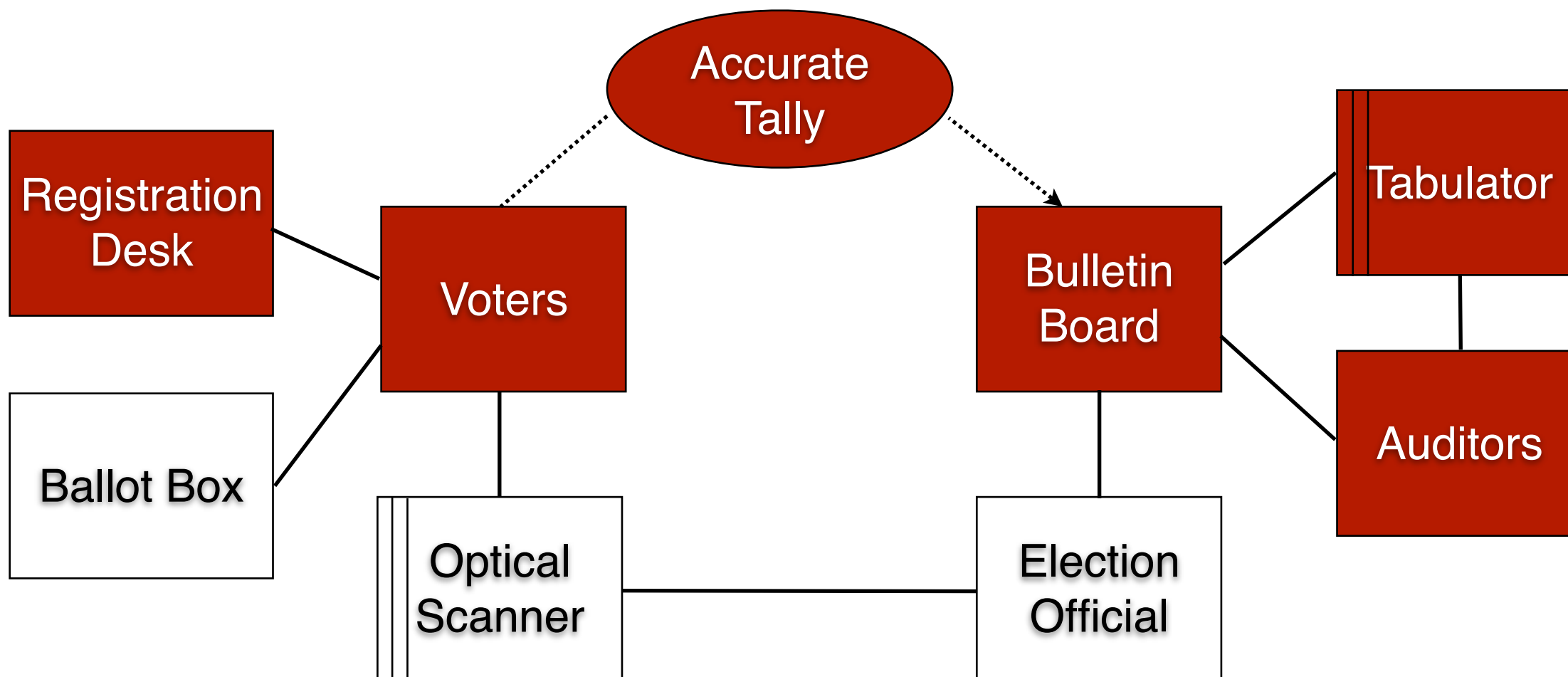
- modeled & analyzed the two designs
- argued why Scantegrity is more dependable



Voting System Designs



Optical Scanner System



Scantegrity

Related Work

Trusted Bases

- trusted computing base (TCB) (Lampson), security kernel (Popek78), separation kernel, safety kernel (Rushby)
- trust assumptions (Haley et. al)

Related Work

Dependability Cases

- Kelly (York), Knight (Virginia), Rushby (SRI), etc.
- Software Certification Consortium (USA)

Requirements Specification

- Problem Frames
- Goal-oriented approaches (KAOS, i^* , Tropos, etc)

Conclusions

Articulate

and prioritize most critical requirements

Design

for small trusted bases, easier arguments

Analyze

your arguments