A Dependability Case Language for a Radiation Therapy System

Michael Ernst, Dan Grossman, Jon Jacky,
Calvin Loncaric, Stuart Pernsteiner,
Zachary Tatlock, Emina Torlak, Xi Wang

University of Washington

end-to-end verification for safety critical systems

Memory Model

COMPCERT

Quark

frenetic >>

IronClad

seL4

Memory Model

COMPCERT

Quark

frenetic >>

IronClad

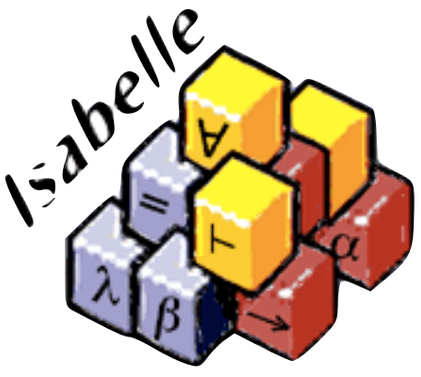
seL4

SUPPORTED BY

SLAM

ROSETTE

alloy



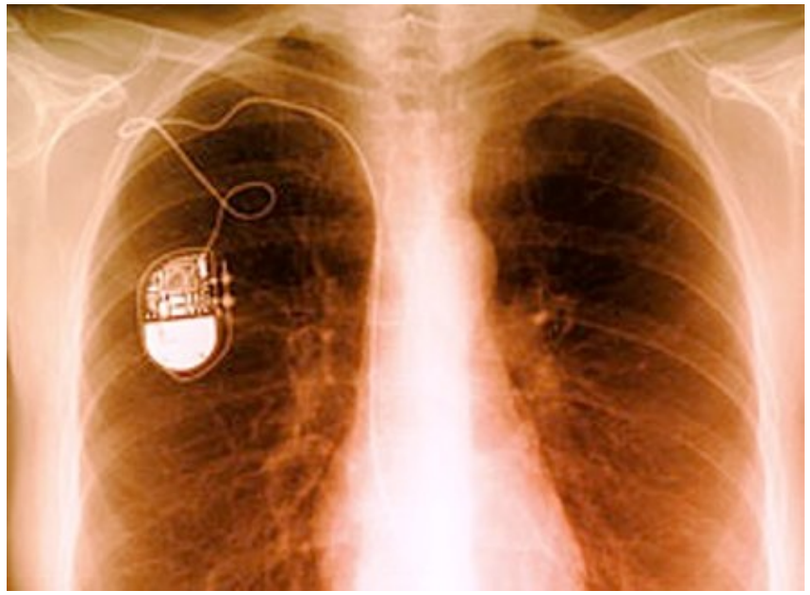
Formal

COMPCERT

Quark

IronClad

frenetic ≫ seL4




SPACEX


NYSE
CYPRESS

Formal

COMPCERT

IronClad

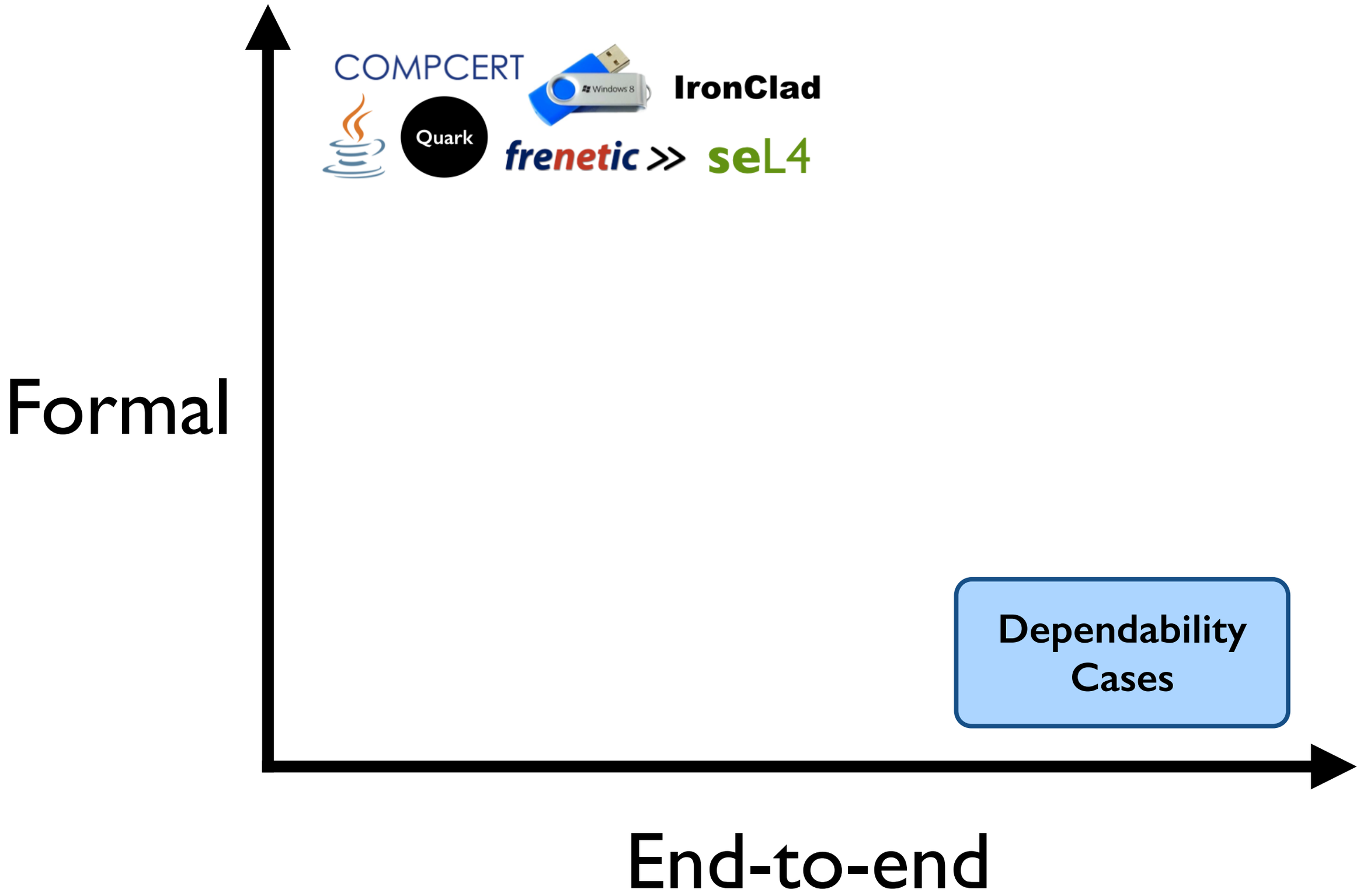
frenetic >>

seL4

Formal

Formal

End-to-end


COMPCERT
Windows 8
IronClad
Quark
frenetic ≫
seL4
Formal
Dependability Cases
End-to-end

Dependability cases

Integrate diverse sources of evidence

check interfaces of design, testing, proof, review

Argue end-to-end claim based on evidence

show claim holds across all layers of a system

Focus on physical system properties

eases validation and focuses verification effort

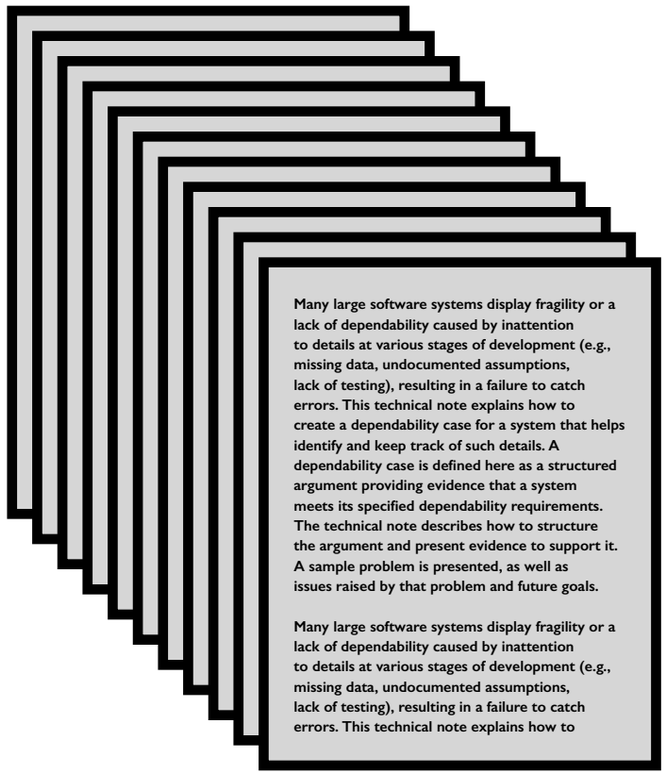
Dependability case engineering

Dependability case engineering

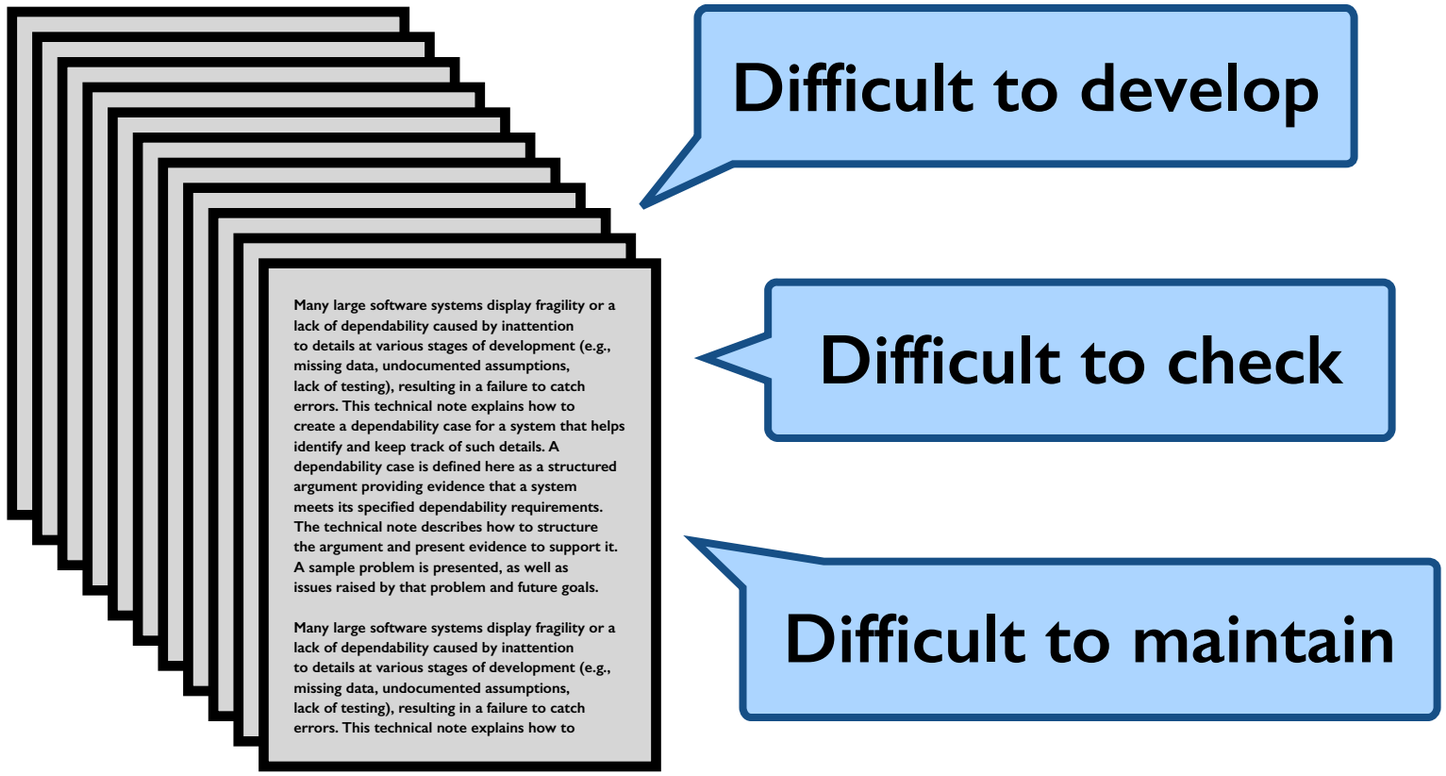
Many large software systems display fragility or a lack of dependability caused by inattention to details at various stages of development (e.g., missing data, undocumented assumptions, lack of testing), resulting in a failure to catch errors. This technical note explains how to create a dependability case for a system that helps identify and keep track of such details. A dependability case is defined here as a structured argument providing evidence that a system meets its specified dependability requirements. The technical note describes how to structure the argument and present evidence to support it. A sample problem is presented, as well as issues raised by that problem and future goals.

Many large software systems display fragility or a lack of dependability caused by inattention to details at various stages of development (e.g., missing data, undocumented assumptions, lack of testing), resulting in a failure to catch errors. This technical note explains how to

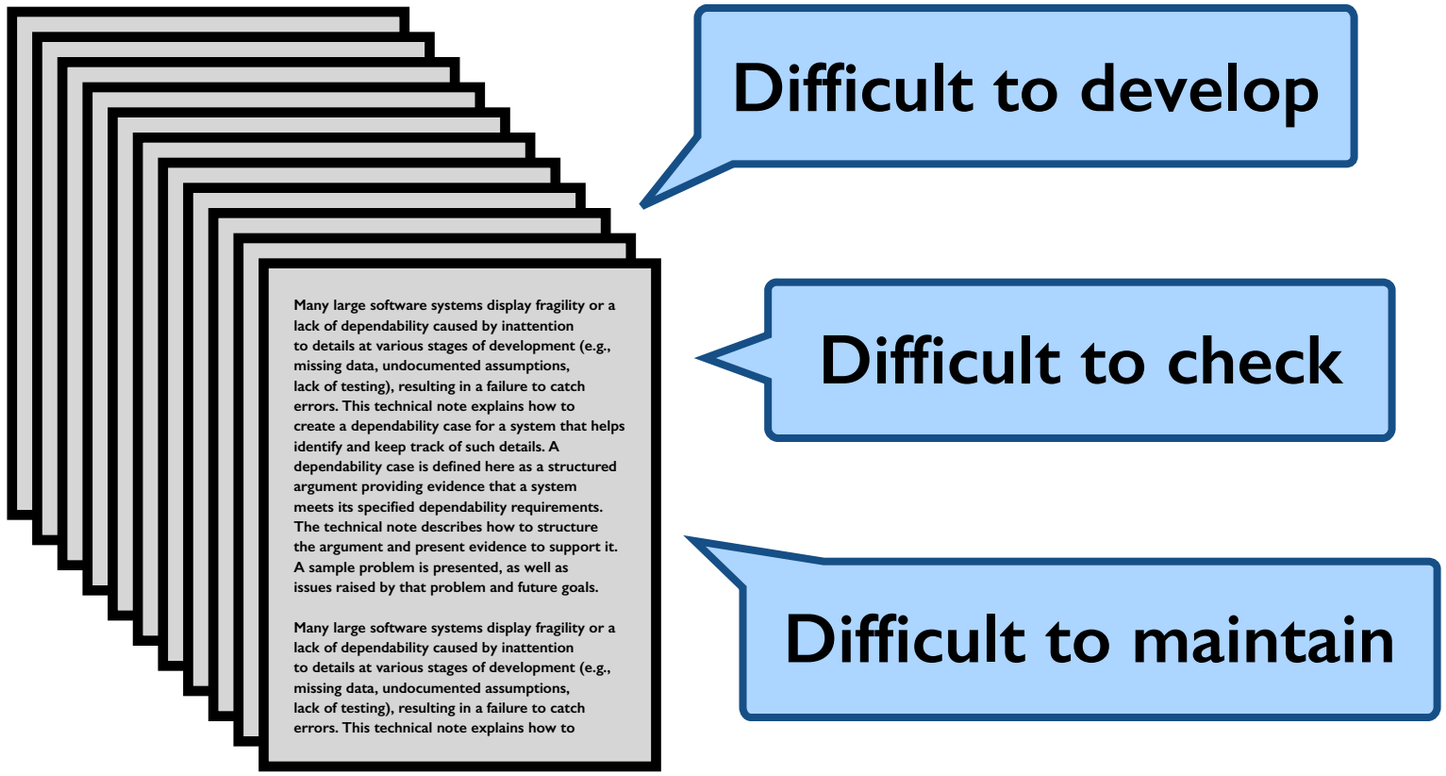
Dependability case engineering



Dependability case engineering

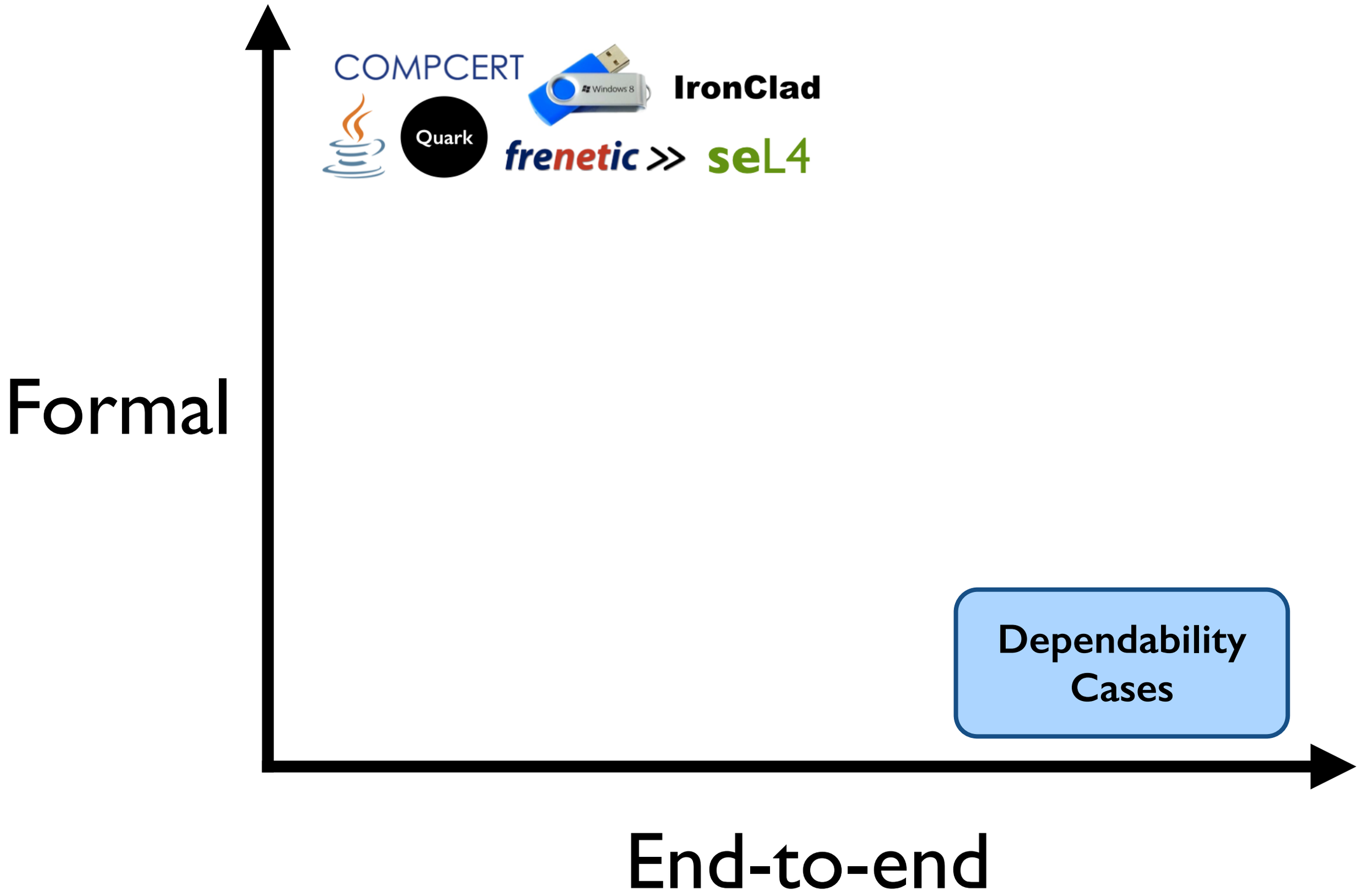


Dependability case engineering

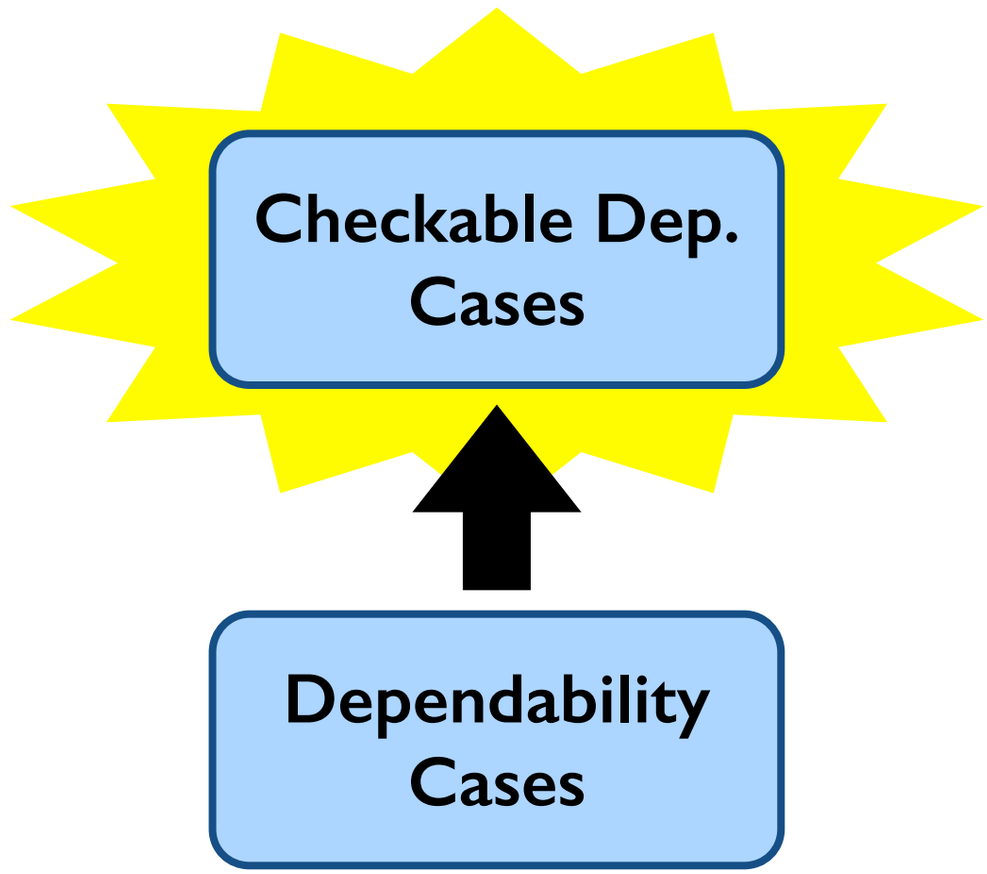


SUPPORTED BY

?


COMPCERT
Windows 8
IronClad
Quark
frenetic ≫
seL4
Dependability Cases
Formal
End-to-end

Formal



End-to-end

Developing a Dependability Case Language

Developing a Dependability Case Language

Move from specific to general

avoid attempt to design "silver bullet"

Developing a Dependability Case Language

Move from specific to general

avoid attempt to design "silver bullet"

1. **Target specific system**

Developing a Dependability Case Language

1. Target specific system
2. Develop dep. claims

Developing a Dependability Case Language

Claims

1. Target specific system
2. Develop dep. claims

Developing a Dependability Case Language

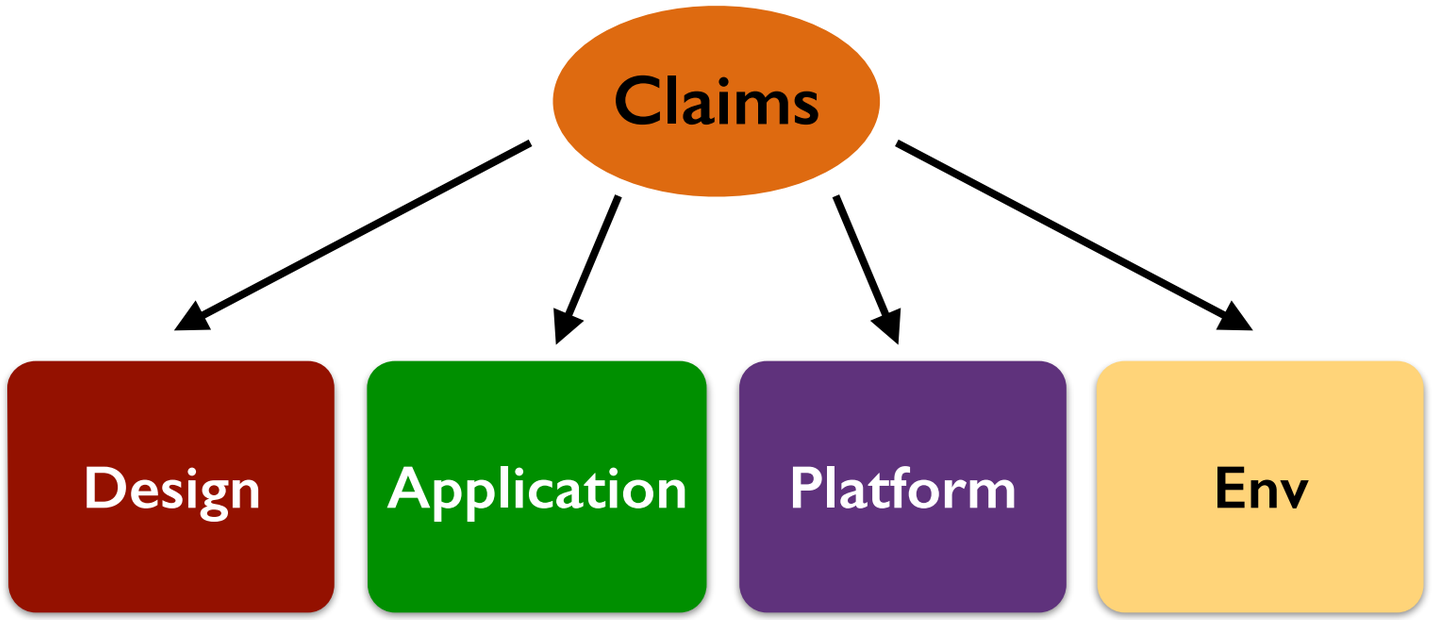
1. Target specific system
2. Develop dep. claims

Developing a Dependability Case Language

Claims

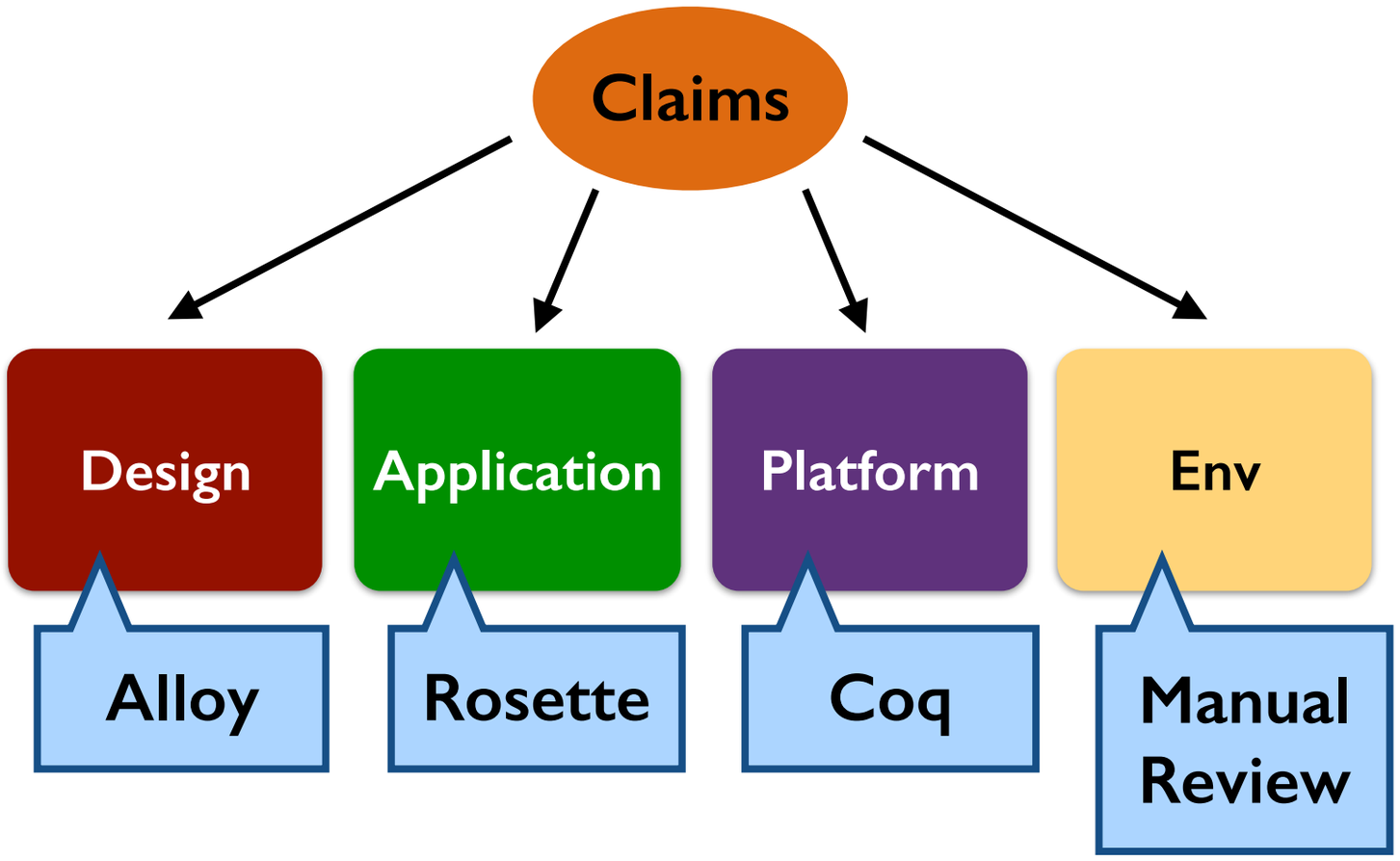
1. Target specific system
2. Develop dep. claims
3. Gather evidence

Developing a Dependability Case Language



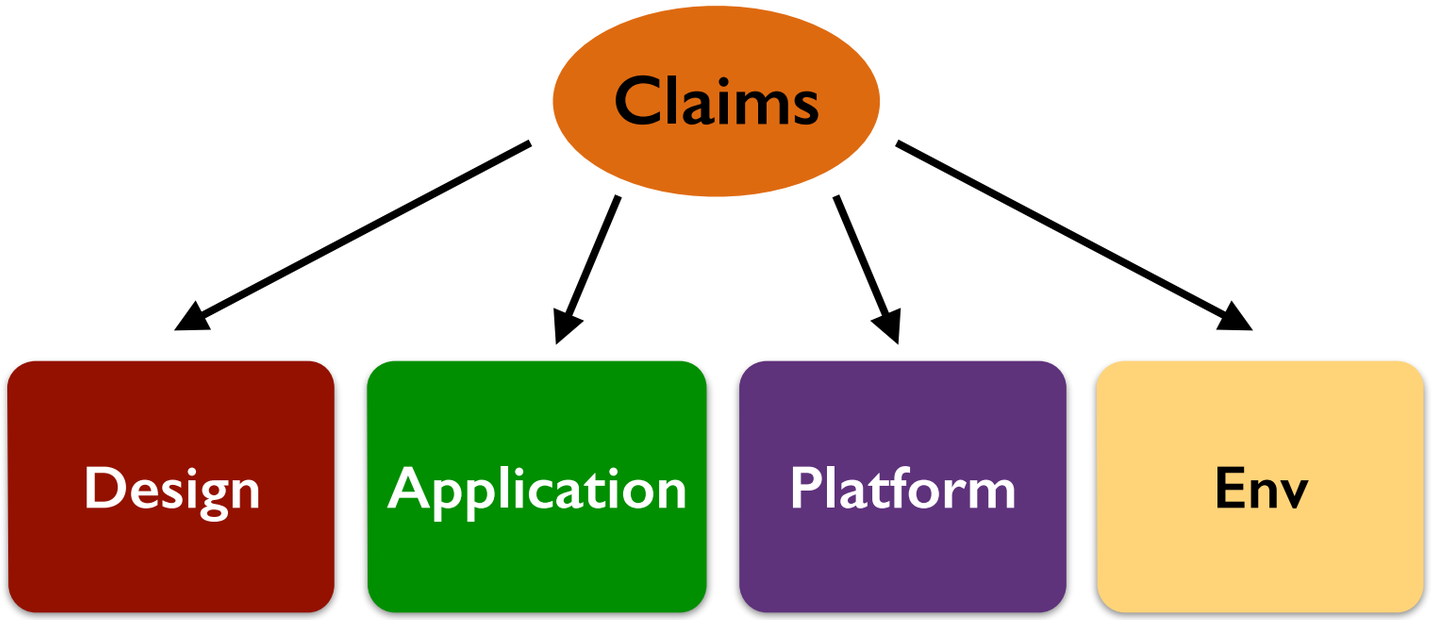
1. Target specific system
2. Develop dep. claims
3. Gather evidence

Developing a Dependability Case Language



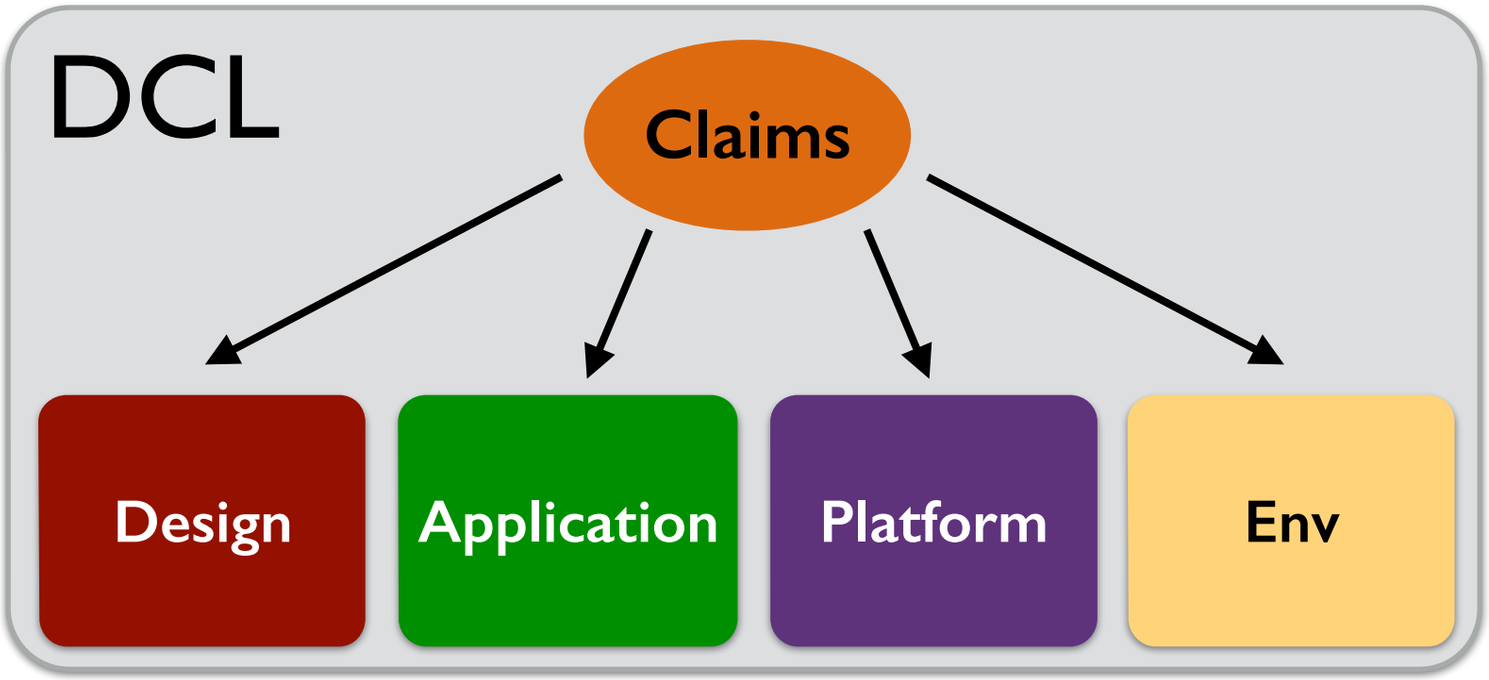
1. Target specific system
2. Develop dep. claims
3. Gather evidence

Developing a Dependability Case Language



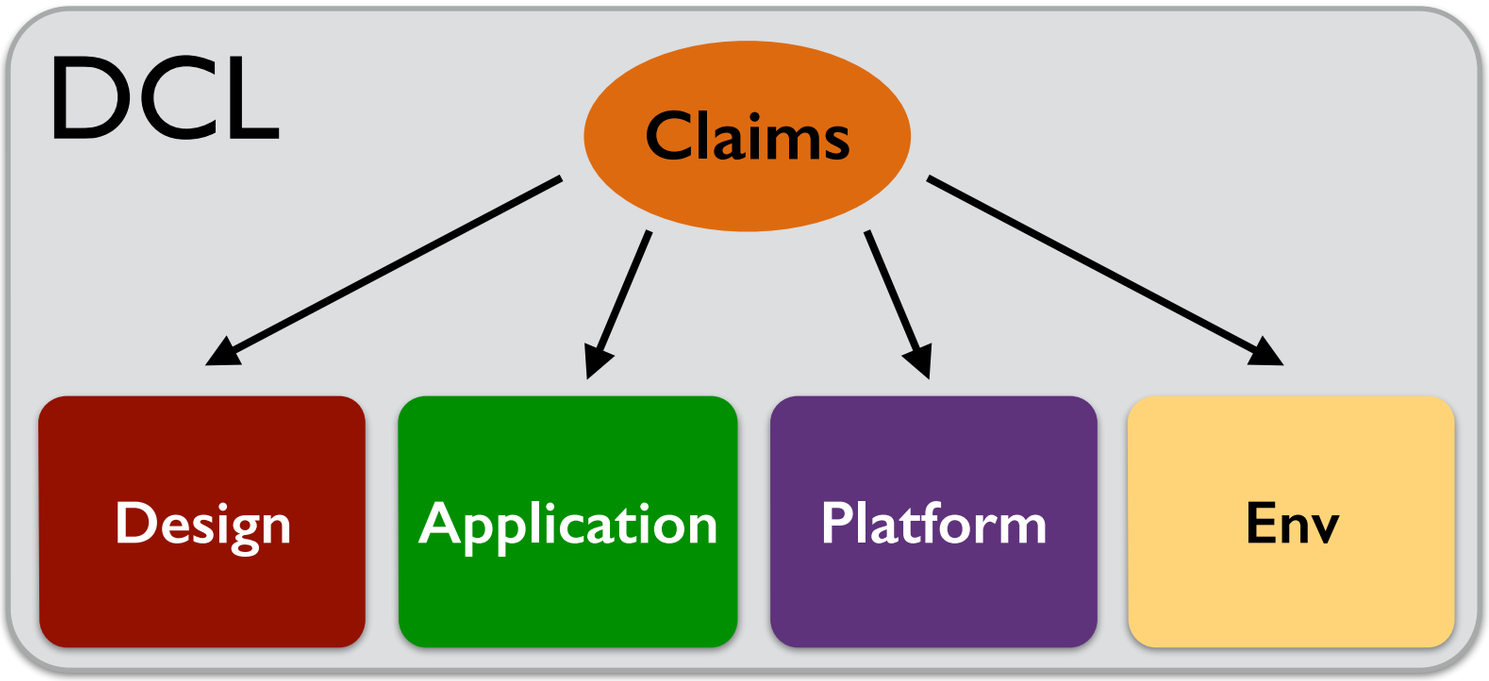
1. Target specific system
2. Develop dep. claims
3. Gather evidence
4. Design + build DCL

Developing a Dependability Case Language



1. Target specific system
2. Develop dep. claims
3. Gather evidence
4. Design + build DCL

Developing a Dependability Case Language



1. Target specific system
2. Develop dep. claims
3. Gather evidence
4. Design + build DCL

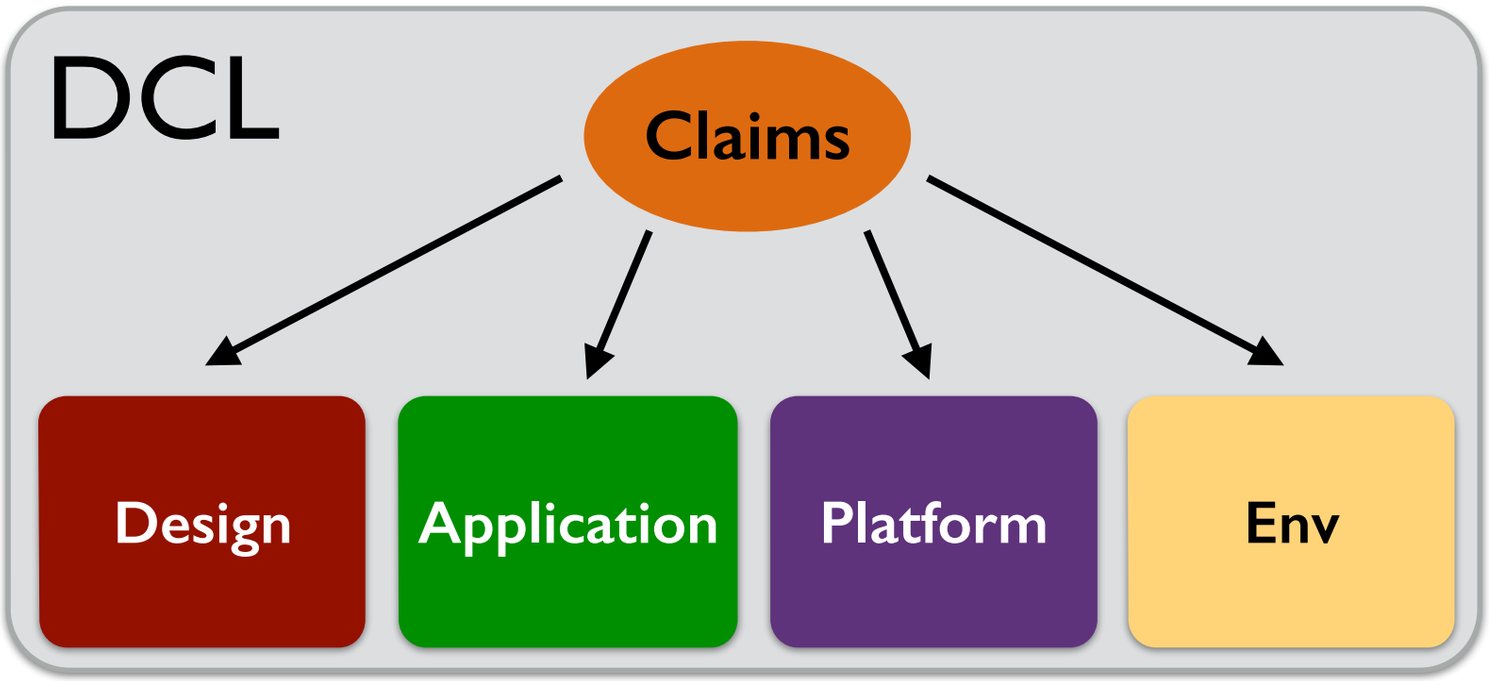
Find general tradeoffs and patterns

make simple easy and hard possible

Impact real-world projects

bring current PL tech to the trenches

Developing a Dependability Case Language



1. Target specific system
2. Develop dep. claims
3. Gather evidence
4. Design + build DCL

Find general tradeoffs and patterns

make simple easy and hard possible

Impact real-world projects

bring current PL tech to the trenches

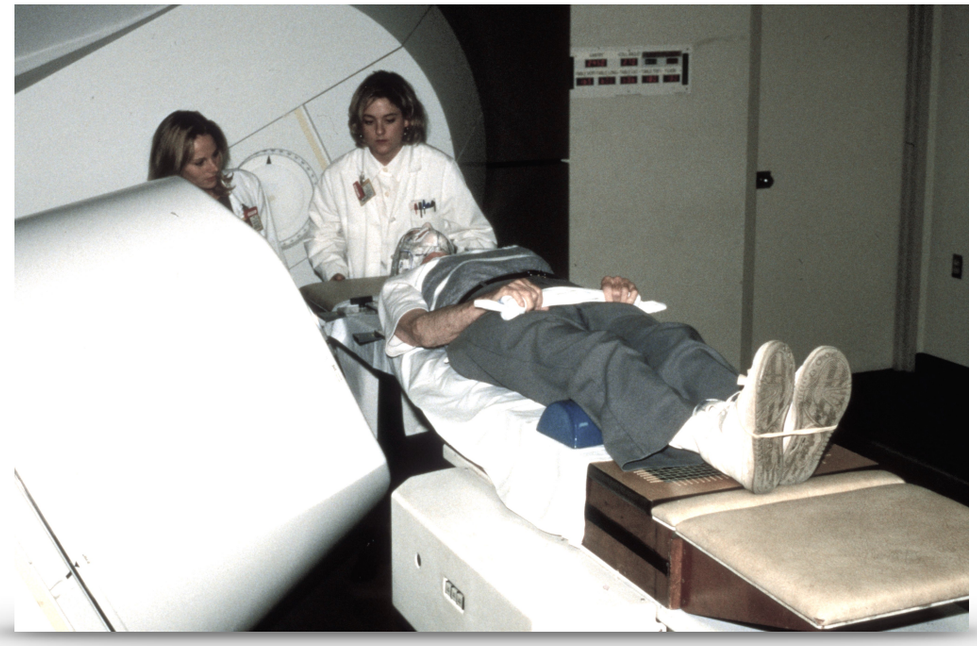
an end-to-end dependability case for CNTS

results

Checking safety of CNTS

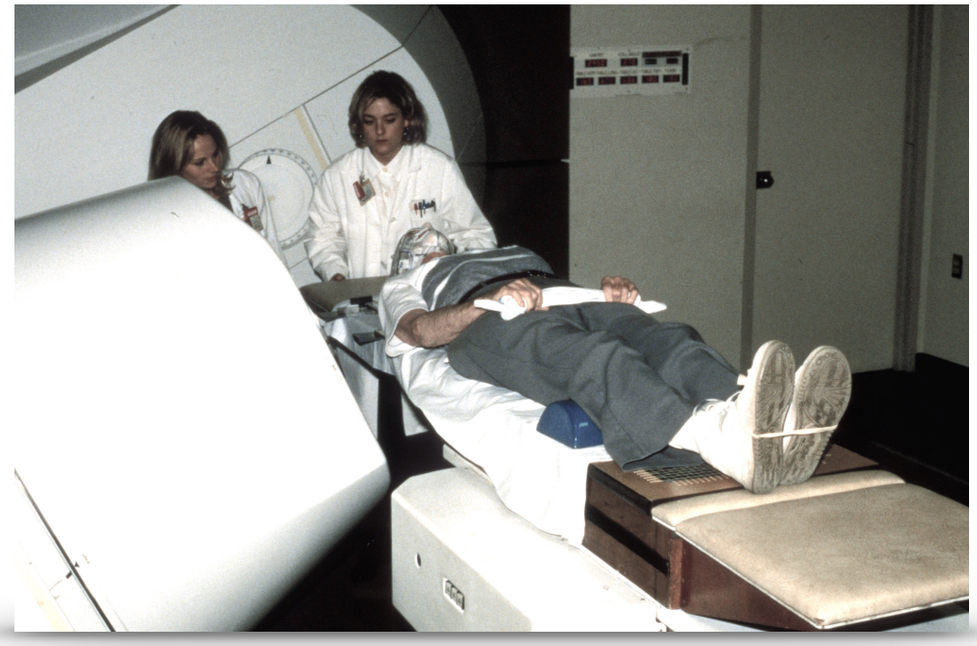
Clinical Neutron Therapy System (CNTS) at UW

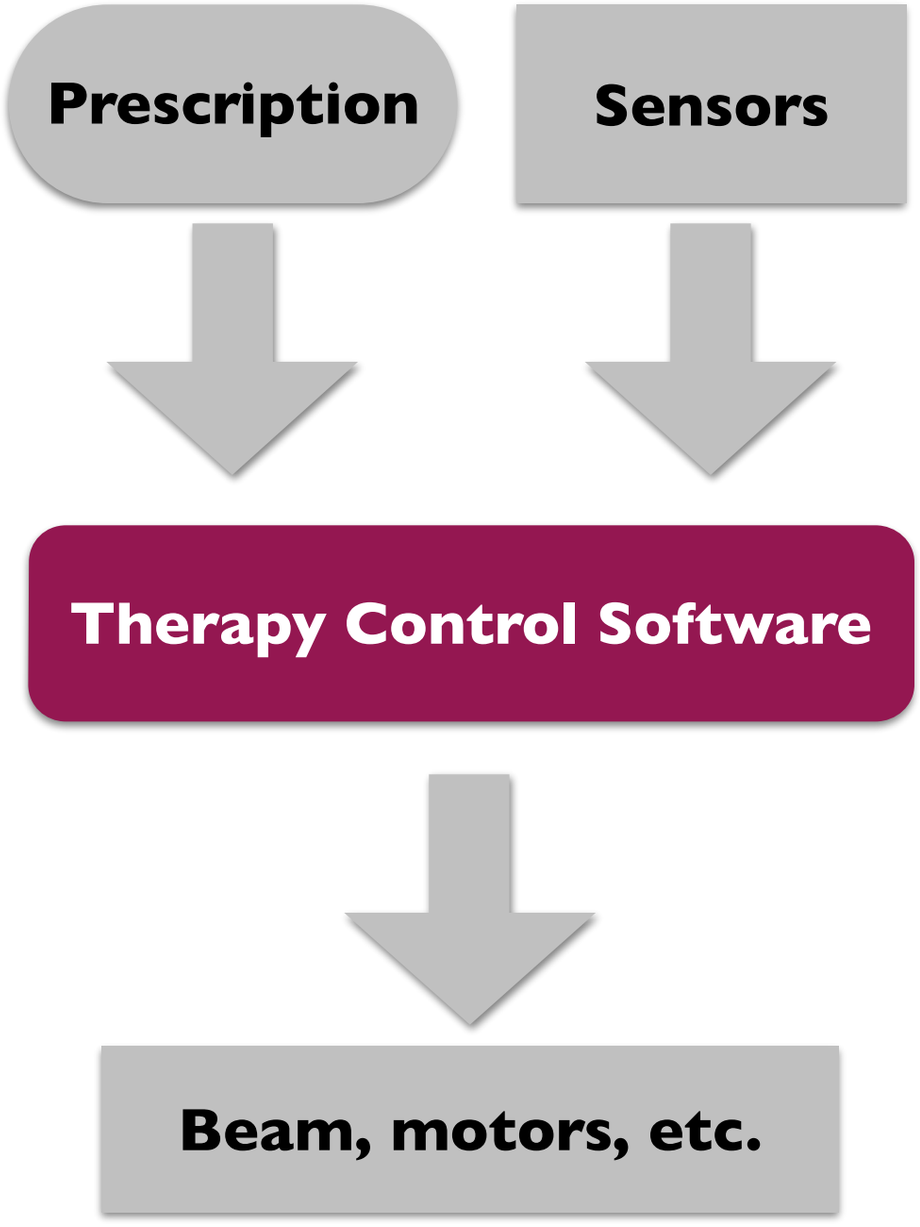
- 30 years of incident-free service.
- Controlled by custom software, built by CNTS engineering staff.
- Third generation of Therapy Control software now being built.



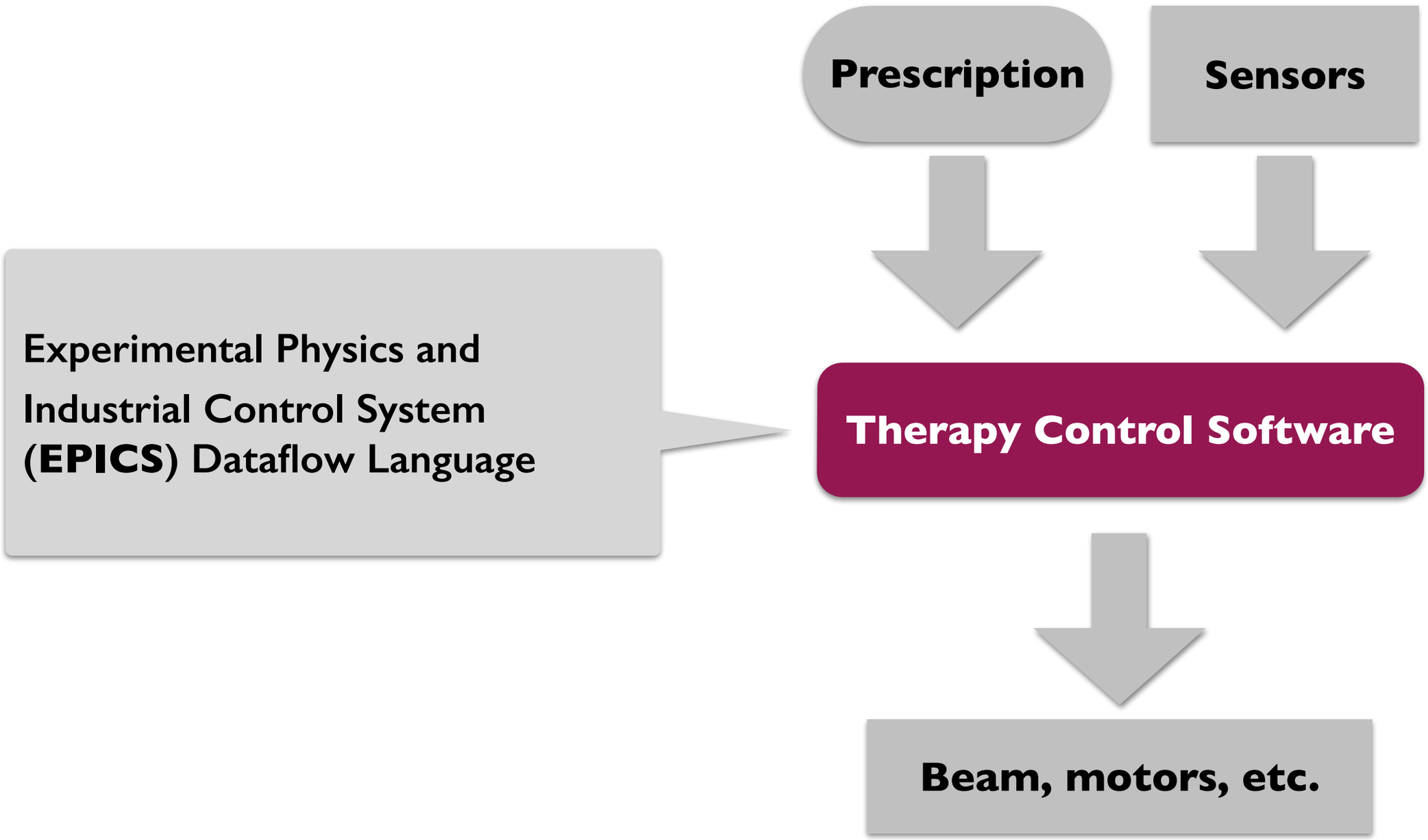
Checking safety of CNTS

Clinical Neutron Therapy System (CNTS) at UW





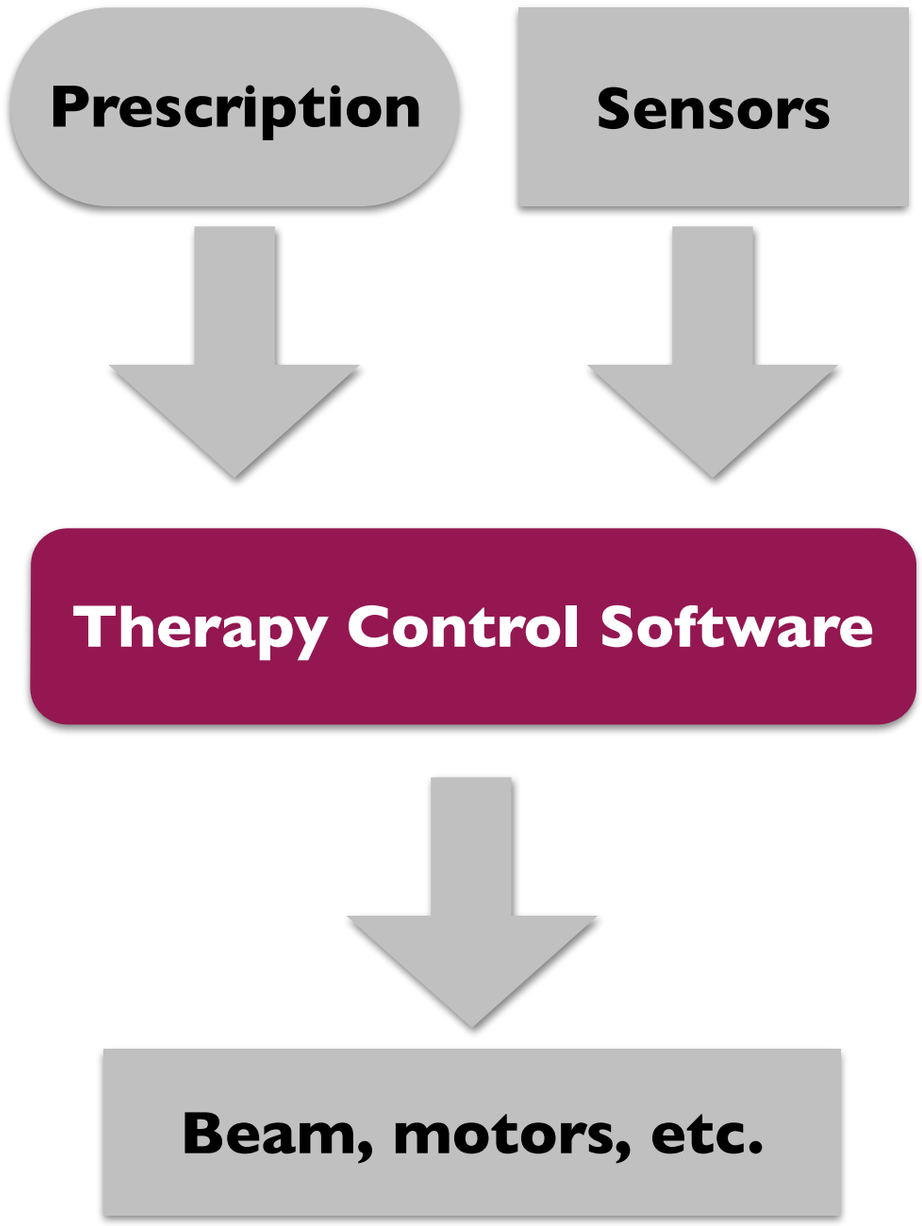
Checking safety of CNTS



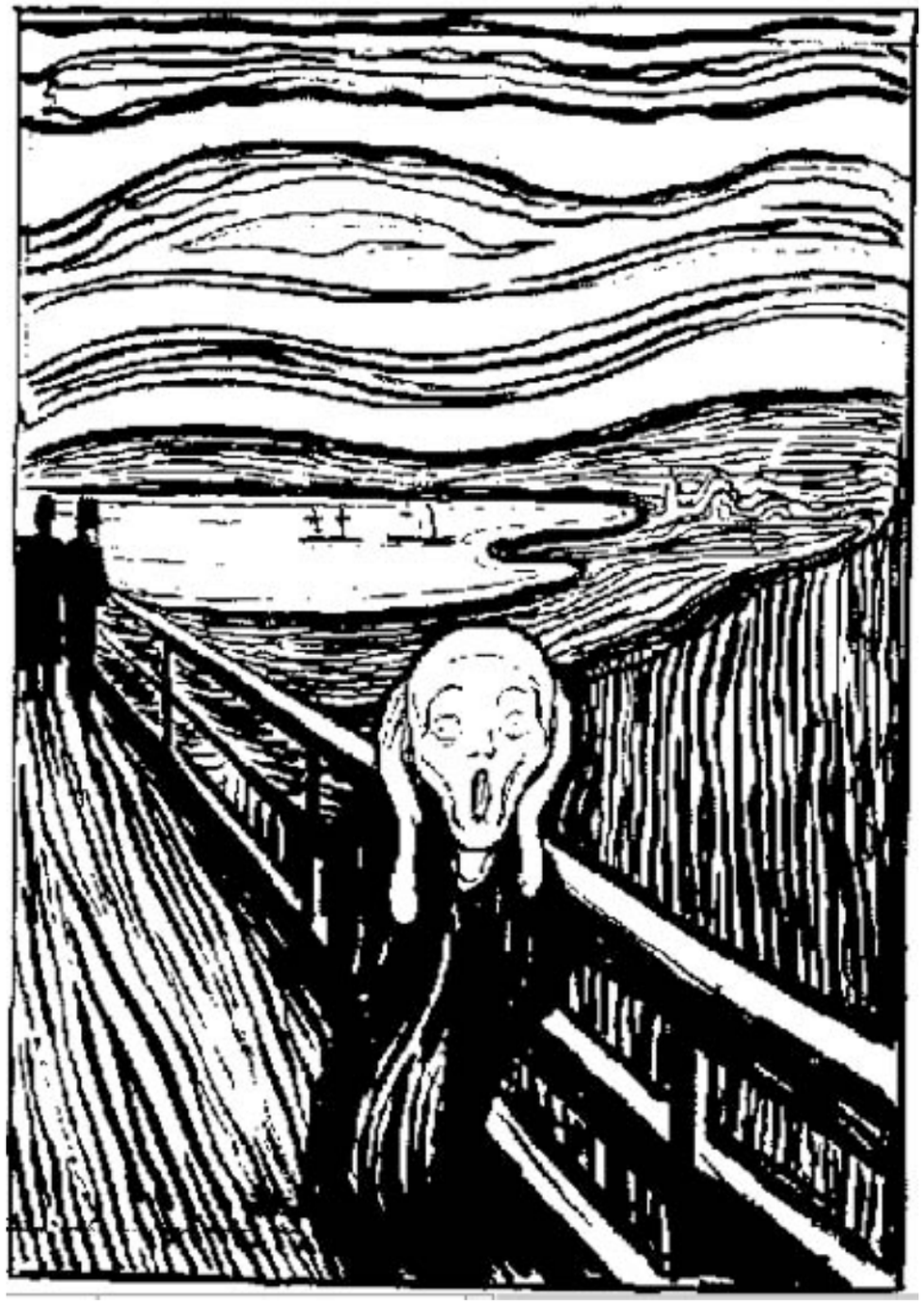
Checking safety of CNTS

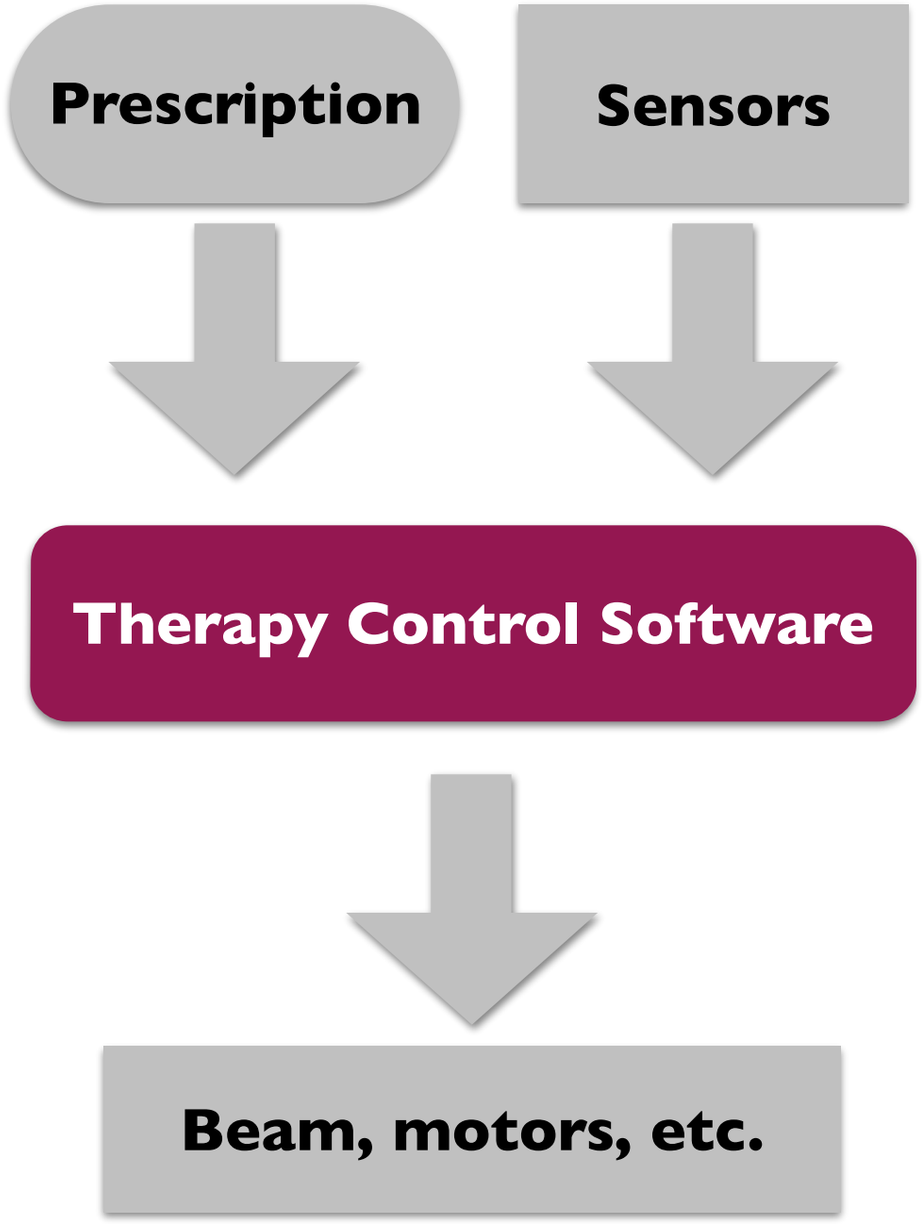
EPICS documentation / semantics

The Maximize Severity attribute is one of NMS (Non-Maximize Severity), MS (Maximize Severity), MSS (Maximize Status and Severity) or MSI (Maximize Severity if Invalid). It determines whether alarm severity is propagated across links. If the attribute is MSI only a severity of INVALID_ALARM is propagated; settings of MS or MSS propagate all alarms that are more severe than the record's current severity. For input links the alarm severity of the record referred to by the link is propagated to the record containing the link. For output links the alarm severity of the record containing the link is propagated to the record referred to by the link. If the severity is changed the associated alarm status is set to LINK_ALARM, except if the attribute is MSS when the alarm status will be copied along with the severity.



Checking safety of CNTS



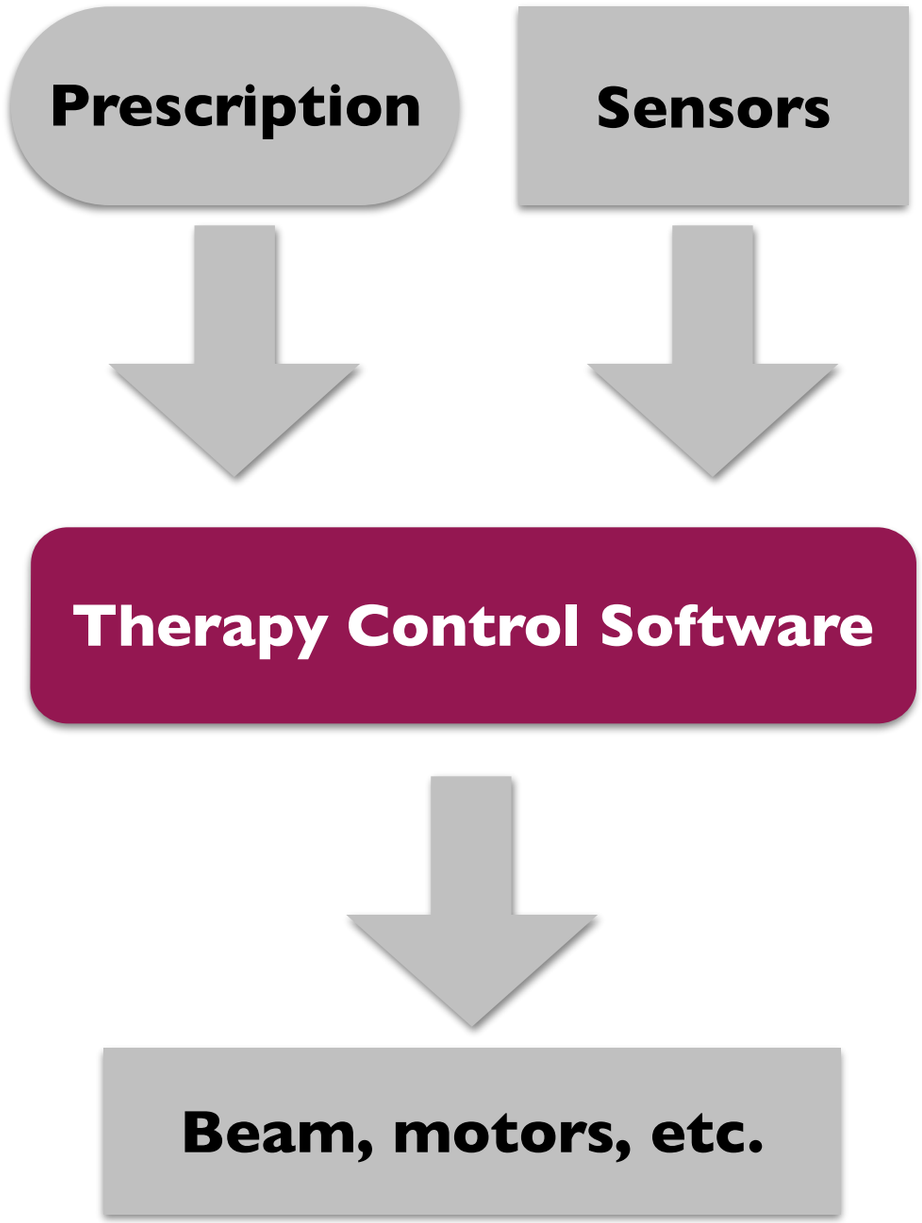


Checking safety of CNTS

An *end-to-end property* that spans the entire system, not just software.

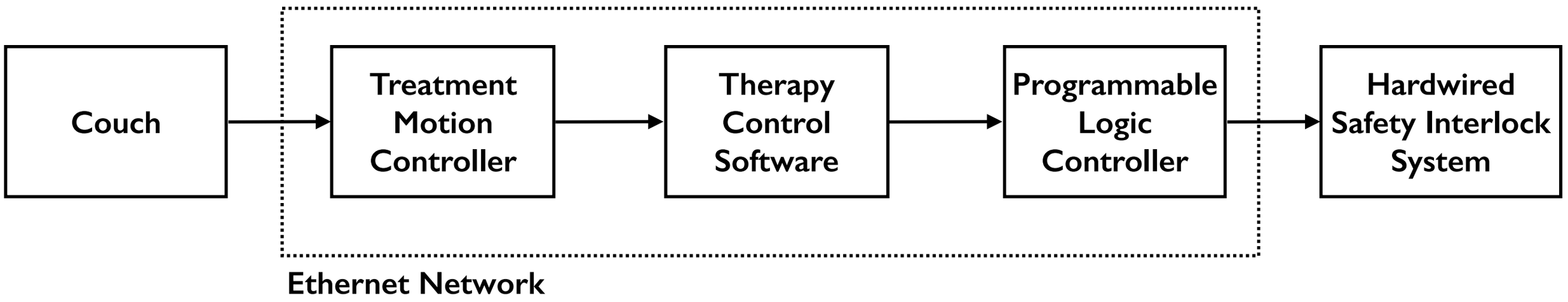
CNTS Couch Safety Property:

The beam will turn off if the couch rotation angle moves out of tolerances during treatment and the operator has not issued the manual override command.

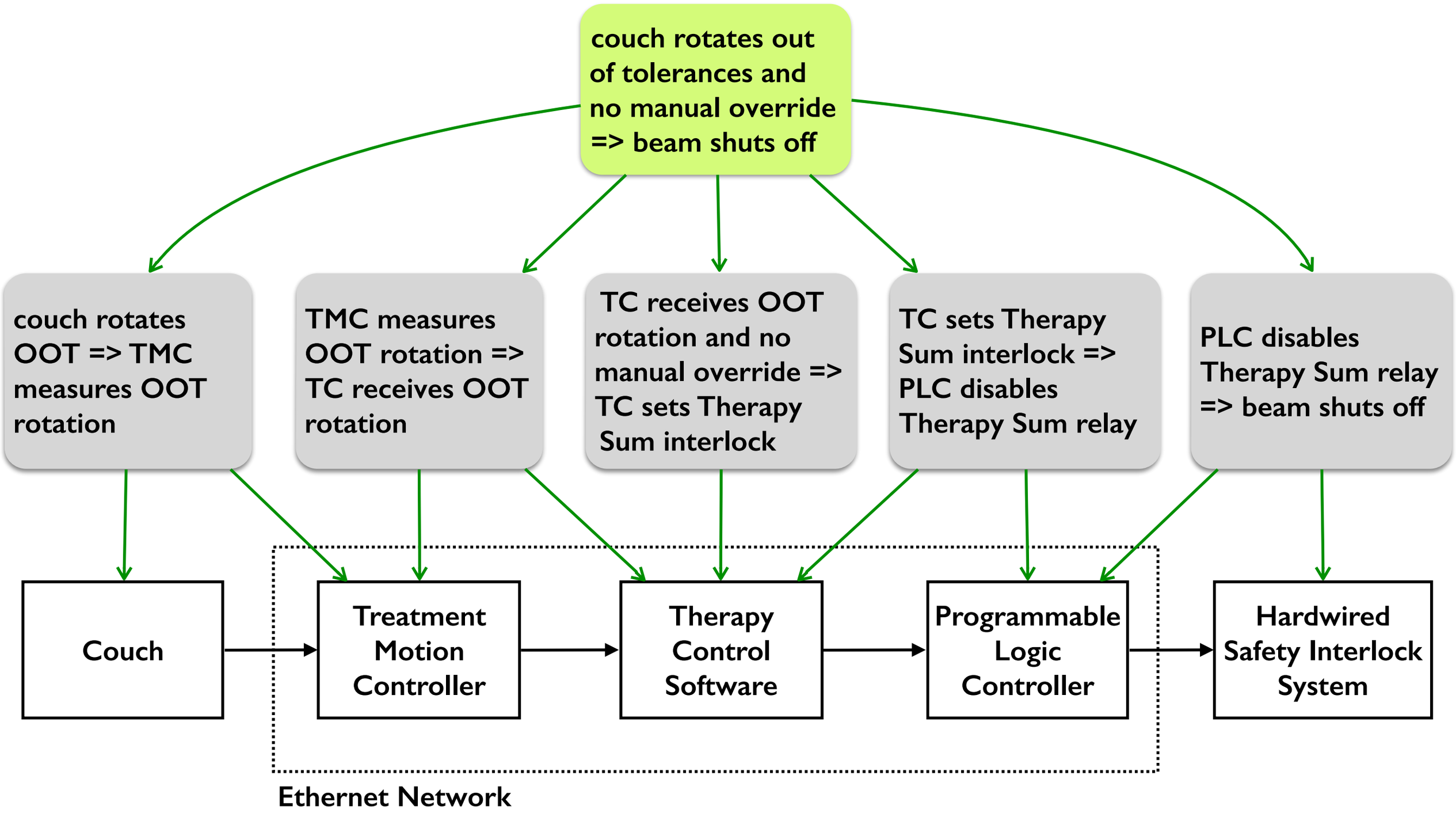


An informal dependability case for couch safety

couch rotates out
of tolerances and
no manual override
=> beam shuts off



An informal dependability case for couch safety



A formal dependability case for couch safety

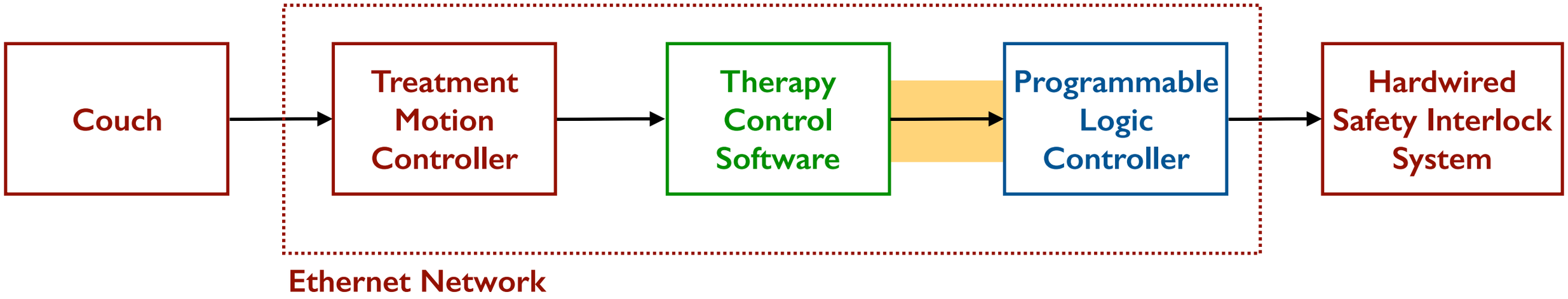
couch rotates out of tolerances and no manual override => beam shuts off

```
all r: Couch.rotation |
  (properties and
    r.angle not in Prescription.tolerance and
    no Event.GantryCouch_Turntable_Override) =>
 some Beam.state & BeamOff
```

PLC disables Therapy Sum relay => beam shuts off

```
evidence[“63c8d380", PLC_Analysis, …, Proof] =>
all relayState: plc.relay2754 & RelayOpen |
  one coilState: plc.sentMsgs & relayState.^next |
    coilState.coilNumber = Coil1623
    coilState.coilValue = False
```

Generating evidence for couch safety



Expert Review Validator

EPICS Linter

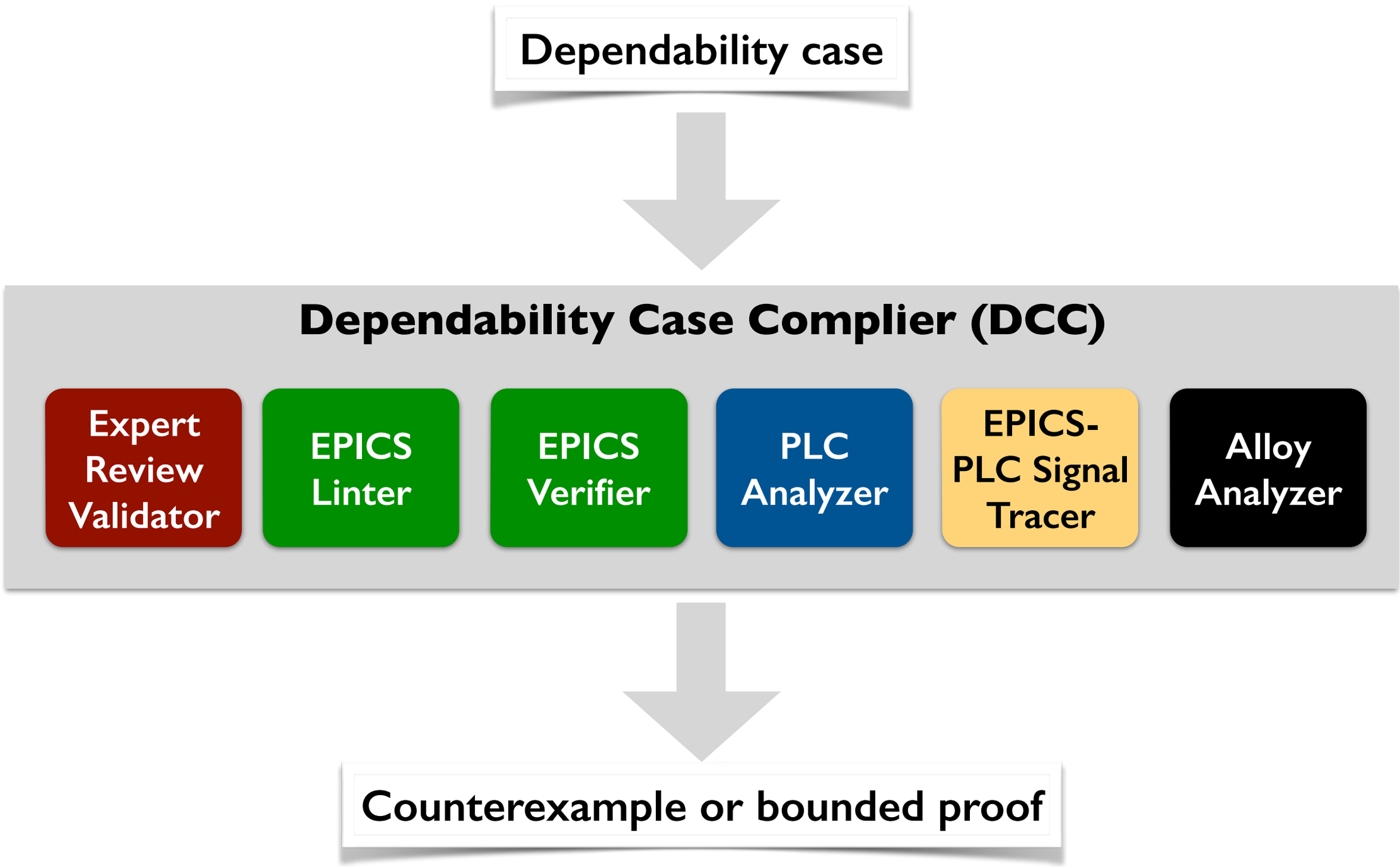
PLC Checker

A *solver-aided verifier* for the subset of EPICS used in CNTS.

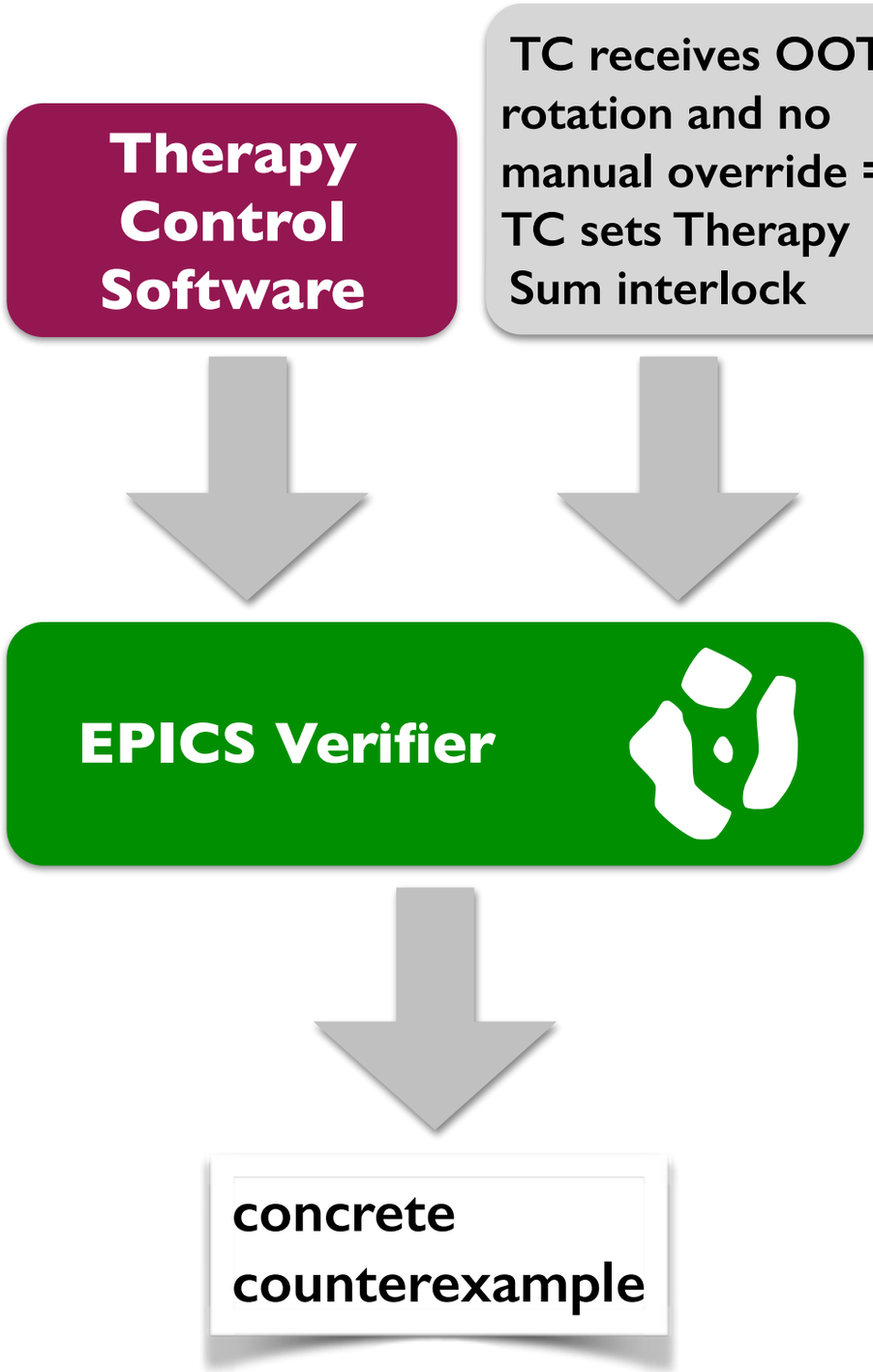
EPICS Verifier

EPICS-PLC Signal Tracer

Checking couch safety

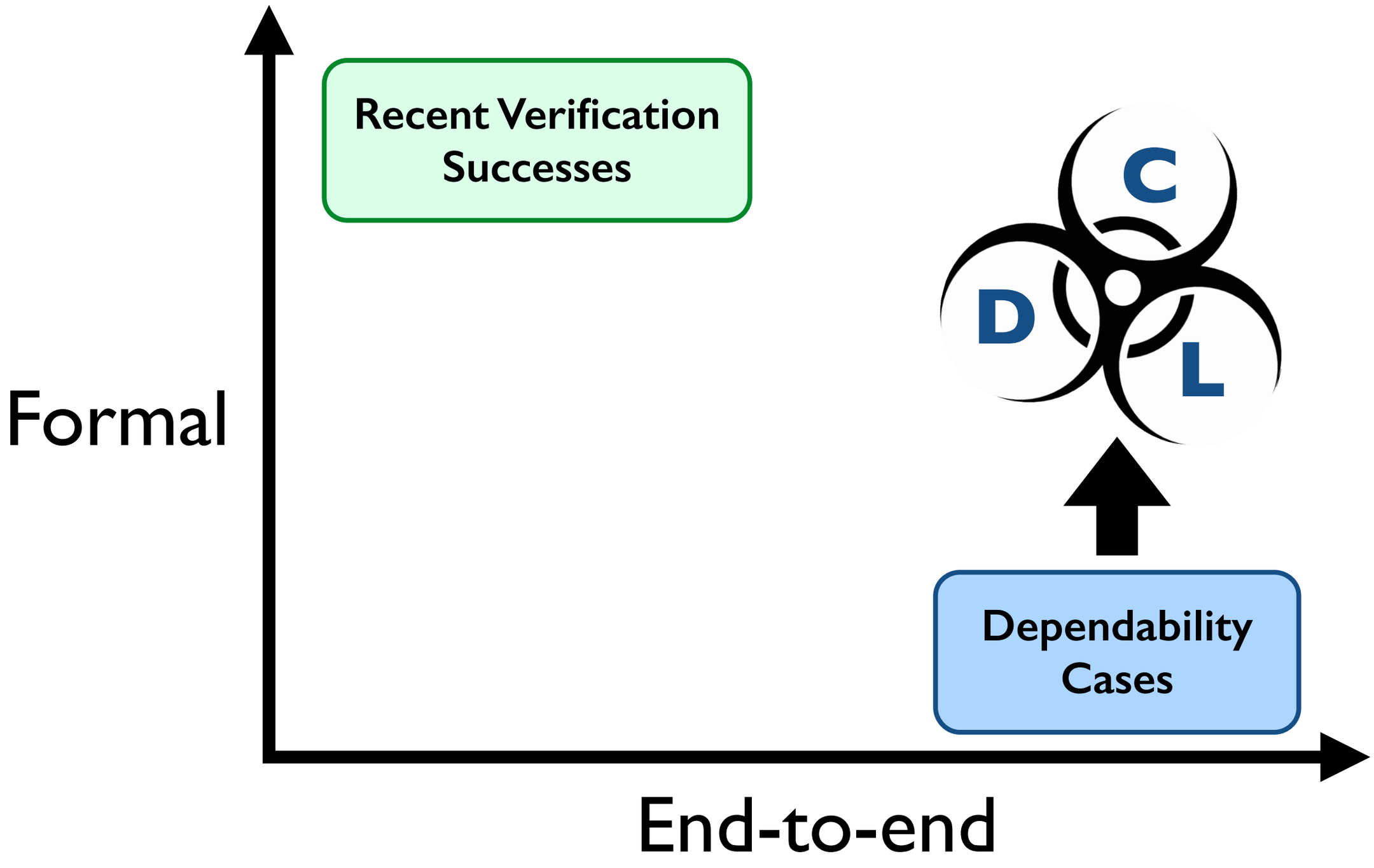


Deep analysis with <2000 LOC of tool code ...



Found a bug in the Therapy Control software (preventing beam shut off), masked by a bug in the EPICS runtime!

Thanks!


Recent Verification Successes
C
D
L
Dependability Cases
Formal
End-to-end