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
Formal

COMPCERT

Quark

IronClad

frenetic

seL4
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

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 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.

SUPPORTED BY

Difficult to develop
Difficult to check
Difficult to maintain

?
Developing a Dependability Case Language

Move from specific to general

avoid attempt to design “silver bullet”
Developing a Dependability Case Language

1. Target specific system
2. Develop dep. claims

Claims
Developing a Dependability Case Language

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

- Claims
  - Design
    - Alloy
  - Application
    - Rosette
  - Platform
    - Coq
  - Env
    - Manual Review
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
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.

Therapy Control Software

Beam, motors, etc.
Checking safety of CNTS

EPICS documentation / semantics

The beams, motors, etc. prescription and sensors are connected by links. If the
severity of an alarm 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.

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 end-to-end property that spans the entire system, not just software.
An informal dependability case for couch safety

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

- couch rotates OOT => TMC measures OOT rotation
- TMC measures OOT rotation => TC receives OOT rotation
- TC receives OOT rotation and no manual override => TC sets Therapy Sum interlock
- TC sets Therapy Sum interlock => PLC disables Therapy Sum relay
- PLC disables Therapy Sum relay => beam shuts off

Couch -> Treatment Motion Controller -> Therapy Control Software -> Programmable Logic Controller -> Hardwired Safety Interlock System

Ethernet Network
A formal dependability case for couch safety

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

- **PLC disables Therapy Sum relay** => beam shuts off

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

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

- Couch
- Treatment Motion Controller
- Therapy Control Software
- Programmable Logic Controller
- Hardwired Safety Interlock System

**Ethernet Network**

**Software Tools**
- Expert Review Validator
- EPICS Linter
- PLC Checker
- EPICS Verifier
- EPICS-PLC Signal Tracer

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

Dependability Case Complier (DCC)

Countereexample or bounded proof
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!

TC receives OOT rotation and no manual override => TC sets Therapy Sum interlock

Therapy Control Software

EPICS Verifier

concrete counterexample
Thanks!

Recent Verification Successes

Dependability Cases

Formal

End-to-end