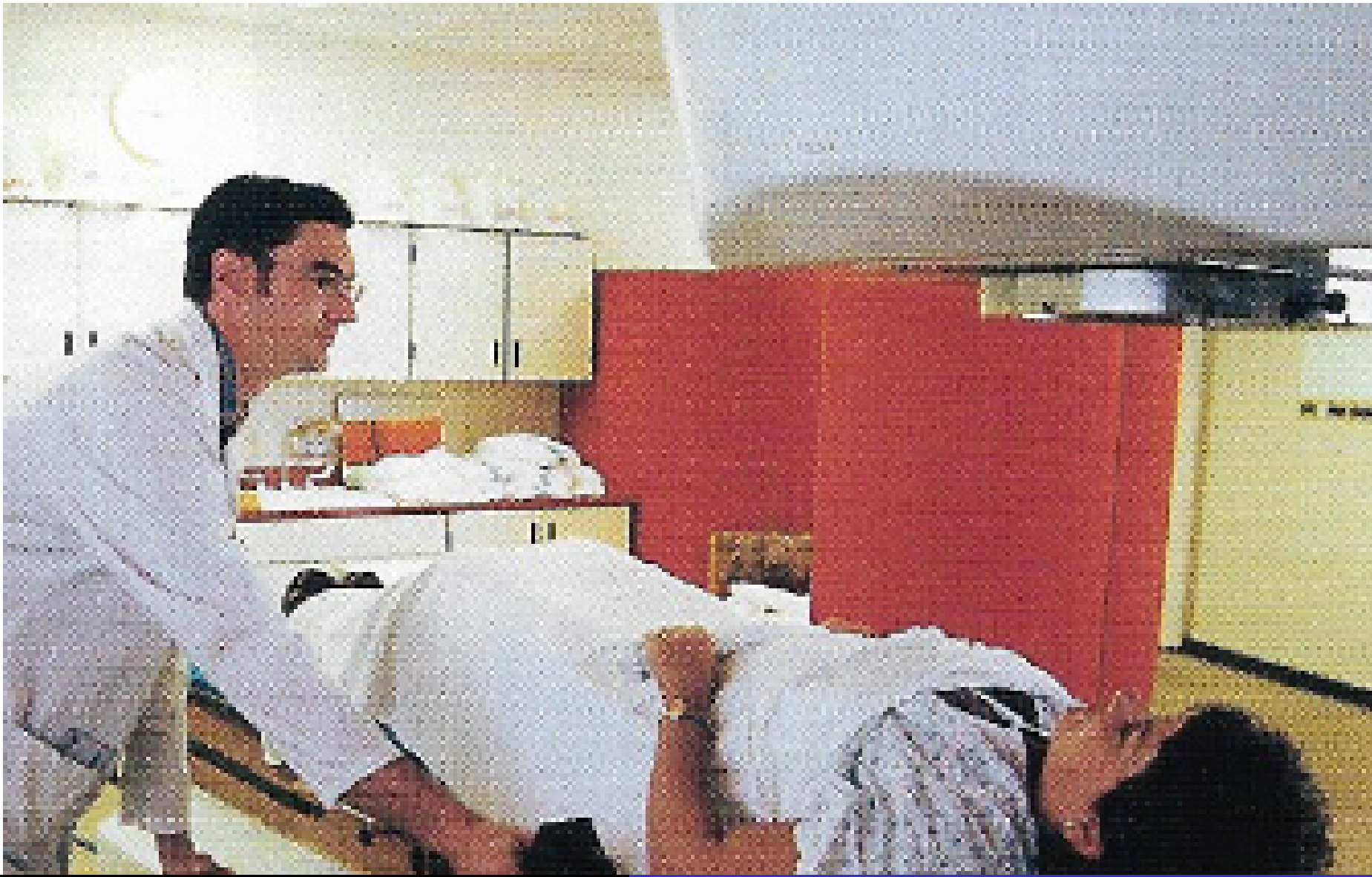
Automatic Formal Verification for EPICS

Jonathan Jacky, Stefani Banerian,
Michael D. Ernst, Calvin Loncaric, Stuart Pernsteiner,
Zachary Tatlock, Emina Torlak

Department of Radiation Oncology
University of Washington Medical Center
Paul G. Allen School of Computer Science and Engineering
University of Washington
jon@uw.edu, http://staff.washington.edu/jon/

UWMC Clinical Neutron Therapy System (CNTS)

Hospital-based cyclotron and neutron radiation therapy



Safety requirements

We must ensure that the we satisfy this overall safety requirement:

- *The neutron beam can only turn on or remain on when the machine setup matches a prescription that has been selected by the operator.*

This overall requirement is composed of hundreds of detailed requirements, for example:

- *The actual gantry angle must match the prescribed angle within a given tolerance, when the machine is in therapy mode and that setting has not been overridden and . . .*

These requirements cannot be checked with relays or PLCs.

General purpose computing in high-level languages is unavoidable.

EPICS and safety-critical systems

Is it advisable to build a safety-critical system with EPICS?
Some conventional wisdom says no:

> 2008: "(EPICS) code is not rigorously audited to the standards . . . that would be needed (for medical applications). . . ."
> `epics/tech-talk/2008/msg00803.php`

Response: We reviewed and tested EPICS code ourselves.

> 2012: "EPICS should never be relied on for safety-critical operations . . ."
> `epics/tech-talk/2012/msg01836.php`

Response: We use a subset of EPICS in a restricted style.

Outline

Using EPICS with confidence in safety-critical applications

Three related topics::

- Selecting an EPICS subset and a restricted programming style
- New tool for finding errors in EPICS databases (applications)
- New tool for testing EPICS core (runtime)

Control Program: Limiting Complexity (1)

EPICS architecture:

- Embedded computers with EPICS runtime, called *IOCs*
- Application programs on IOCs, called *databases*
- User interface, database access etc. provided by separate *Client* programs running on different computers
- IOCs and clients communicate using *Channel Access* (CA) network protocol

Therapy control application architecture:

- Therapy control program runs on one IOC
- Therapy IOC alone executes control laws, achieves and maintains safe state
- Clients and CA are only used to make progress. For example, select and load prescription
- Do not depend on clients or CA for control laws or safety

Clients and CA not considered in our formal analyses and tools

Control Program: Limiting Complexity (2)

Select an EPICS subset and use a restricted programming style.

Just enough to support data flow from inputs to outputs:

- Only database records, StreamDevice .proto files, st.cmd
- No SNL, no subroutine records, no custom device support
- Database DB links only, no CA links
- Data flow is all "push" from inputs to outputs:
  SCAN PASSIVE, INPx NPP, OUT PP, FLNK
- 19 record types: `acalcout` `ai` `ao` `asyn` `bi` `bo` `calc` `calcout` `dfanout` `fanout` `longin` `longout` `mbbo` `scalcout` `seq` `stringin` `stringout` `subArray` `waveform`

Our formal analyses and tools only consider these.

EPICS Symbolic Interpreter

The *EPICS Symbolic Interpreter* is a new tool for finding errors in EPICS databases (application programs).

Similar role to unit testing, but considers all possible input values.

Inputs:

- EPICS database: *.db .substitutions .template st.cmd*
- Property that relates PVs before and after processing: *actual gantry* $\neq$ *prescribed gantry* $\Rightarrow$ *interlock set*

Output: *everything is ok!* – property is satisfied – or:

- Counterexample: PVs with values that violate property – *Iso:GantryCouch:Gantry:Prescribed.VAL* $= 312$ *Iso:GantryCouch:Gantry:Actual.VAL* $= 48 \ldots$
- Log: processing along counterexample data flow path

Now in use, found serious errors missed by reviews and testing

EPICS symbolic interpreter (2)

Similar role to unit testing, *but* —

Testing:

- You try to guess an input value that violates the assertion.
- Passing tests are *not conclusive*, a different value might have failed.

Symbolic interpreter:

- Tool finds an input value that violates the assertion, if there is one.
- Verified properties are *conclusive*, all possible values are checked.

The symbolic interpreter can check all values because it considers CALC fields as symbolic formulas.

EPICS Trace Verifier (1)

The *EPICS Trace Verifier* is a new tool for testing the EPICS runtime.

Does the EPICS runtime behave as we expect?

The trace verifier uses a *formal semantics* we derived from the EPICS Record Reference Manual (RRM).

The formal semantics is a new implementation of EPICS in a specialized programming language.

The trace verifier checks if a sample of IOC behavior is consistent with our formal semantics.

EPICS Trace Verifier (2)

We compare of three things:

$$\text{EPICS RRM} \overset{?}{=} \text{Formal semantics} \overset{?}{=} \text{EPICS code}$$

manual (review RRM) — automatic (trace verifier)

We can revise the formal semantics.

We observe discrepancies found by the trace verifier.

EPICS Trace Verifier (3)

Checked traces from over 20 million randomly generated IOCs

No crashes, no outright errors where RRM makes a false statement.

Found two kinds of discrepancies:

- Our misreadings of the RRM
- Omissions, ambiguities in the RRM

Example omissions, ambiguities:

- *dfanout* (etc.) *OMSL* default is *supervisory*, not *closed_loop*
- *calcout* with *DOPT* = *On Change* considers $inf \neq inf$, contrary to IEEE754
- *seq* uses callbacks even if all delays are zero. Callbacks might interleave with other records processing.
- *calc* etc. write from *INPA* to *A* before processing *INPB* (etc.). Observable if later input links refer to earlier fields (etc.)

None of these omissions or ambiguities affect our control program.

Conclusions

We can use EPICS with confidence in safety-critical applications.

We have demonstrated —

- Successful use of restricted EPICS in a safety-critical application
- Tool that finds errors in EPICS databases using exhaustive analysis
- Testing of the EPICS runtime against the RRM finds ambiguities and omissions, but no crashes or outright errors.