Determining Software Safety in Critical Systems

Heidy Khlaaf

Adelard
About Me

Consultant

**Formal Verification**
Evaluating, designing, specifying, and verifying safety-critical systems

**Safety & Security Assessments**
Carrying out safety and security assessments of clients’ systems

**Production of Standards**
Production of standards and guidelines for safety and security related applications and their development
Comp Sci PhD at UCL 2017– Temporal logic model-checking and verification of software systems.

As a PhD student, I collaborated with Microsoft Research Cambridge to create and extend the T2 tool to support temporal property verification.

Focused on verifying Windows device-drivers in hopes of preventing a multitude of errors.
Safety-Critical Systems

What are they?
Systems whose failure or malfunction may result in the following outcomes: death or serious injury to people, loss or severe damage to equipment/property, and environmental harm.

Largest Sector

- Aviation
- Nuclear Energy
- Automotive & Rail
- Defense, Medical, Finance, Autonomous Systems
**Definition**
The process of establishing whether a system satisfies some requirements (properties), using formal methods of mathematics.

**Model**
Formally model a system’s implementation in the appropriate framework.

**Specify**
Formally specify a system’s requirement in an expressive specification language.

**Model-Check**
Exhaustive exploration of a system’s mathematical model against its specifications.

**Analyze**
If not all specifications were satisfied, analyze findings and their consequences.
Industrial formal verification tools are adept in finding thousands of bugs on millions of lines of code.

Invested in producing more nuanced and expressive formal frameworks, despite lack of scalability and applicability.
How are we not dead yet?

01. How software in safety-critical systems is assured.

02. Verification of a specific type of component that is one of the biggest challenges in the nuclear industry – smart devices.

03. Lessons learned from standards, guidelines, and processes used in safety-critical verification and assurance.
Verifying Nuclear Power Plants

Pressurized Water Reactor
Smart Sensors

Embedded devices

Written in C & Assembly
• Can include FPGAs
• No underlying OS
• Interrupt driven (e.g., time-triggered architecture)
• Use of pointers
• Communication protocols

Use of compilers vary depending on processor and hardware

Not written to be verified
• Specification may not be available

10k, 100k, 1000k, 10000k, ... LoC
Embedded applications: reading and writing port data, setting timer registers and reading input captures, etc.

ARM C51 has special data types sfr and sbit that customise the compiler to the target processor:

```c
sfr P0 0x80
sfr P1 0x81
sfr ADCON 0xDE
sbit EA 0x9F
...
timer0_int() interrupt 1 using 2 {
    unsigned char temp1 ;
    unsigned char temp2 ;
    ADCON = 0x08 ; /* Write data to register */
    P1 = 0xFF ; /* Write data to Port */
    io_status = P0 ; /* Read data from Port */
    EA = 1; /* Set bit(enable all interrupts)*/
}
```
Why Smart Sensors?

**Pure analogue sensors disappearing**

**Improved functionality**
- Microprocessors or micro-controllers
- Better accuracy, calibration, diagnostics
- Configurable but not programmable

**Industrial embedded devices**
- Commercial-Off-The-Shelf (COTS)
- Perform a defined function
- Examples include
  - Temperature transmitters
  - Pressure transmitters
  - Voltage regulators
Verifying Nuclear Power Plants

26th Meeting of IAEA Technical Working Group on Nuclear Power Plant Instrumentation & Control (TWG-NPPIC)
IAEA, Vienna, 24 -26 May 2017

UK Report: Status of NPPs & Issues arising from Assessment of Computer Based Safety Systems

Steve Frost
Superintending Nuclear Inspector:
Professional Lead EC&I
Verifying Nuclear Power Plants

**Issues arising from assessment of computer-based safety systems**

- Issues and challenges focussing on areas that are important in a technical context:
  - I&C architecture design
  - Development of coherent safety cases
    - Justification of smart devices
  - Security of computer based systems important to safety
  - Management of I&C ageing (not just in relation to computer-based systems)

*These issues and challenges are interrelated and are common findings across I&C activities in relation to safety systems*
Demonstration of excellence in all aspects of production from the initial specification through to the finally commissioned system

Independent and thorough assessment of a safety system’s fitness for purpose

Weaknesses that are identified and are to be compensated for

ONR SAPs
The primary principles that define the overall approach for nuclear installations in the UK

Production Excellence
ICBM
Compensation Measures

Justification of use of software
Production Excellence

- Application of technical design practice consistent with current accepted standards for the development of software for computer-based safety systems (e.g., IEC 61508, IEC 61513, ISO 26262, DO 331)

- Implementation of a modern standards quality management system (e.g., ISO 9001, Github, ClearQuest, ClearCase, etc.)

- Application of a comprehensive testing program formulated to check every system function
Independent Confidence-Building Measures

- Complete and diverse checking of the production software by a team that is independent of the systems suppliers (e.g., formal verification, static analysis)

- Independent assessment of the comprehensive testing program covering the full scope of the test activities
Verifying Smart Sensors

- Functionality, accuracy, timing, failure integrity, non-interference
- Data-races, deadlock freedom, stack/buffer overflow, memory safety
- Coding standards, complexity metrics

Property Based Approach

Standards Compliance

Safety Justification

Vulnerability Assessment
Verifying Smart Sensors

Industrial verification tools lack property-driven techniques

State-of-the-art formal verification techniques

- Assume standard C compilers (GCC/LLVM)
- Assume standard architectures (Intel x86)
- Assume underlying OS (scheduler, memory management, etc.)
- Little support for embedded devices
- No support for interrupt-driven systems
- Not scalable beyond 10k loc
Even with automated tools, manual inspection and judgement is required
- Are the bugs false positives?
- Are they relevant to the safety and security of the system?
- Which class of bugs are more critical than others?
- What is the probability of the actual bug occurring?
- What is an acceptable level of failure?
- Can testing and field data demonstrate infeasibility of a bug?
- What efforts are required to repair a system?
- Do the repairs themselves pose a safety and security threat? What is the impact?

Hardware verification, specification correctness, standards compliance
**Why Should You Care?**

- **Availability** - readiness for correct service
- **Reliability** - continuity of correct service
- **Safety** - absence of catastrophic consequences on the user(s) and the environment
- **Integrity** - absence of improper system alteration
- **Maintainability** - ability for a process to undergo modifications and repairs
Opening a Dialogue

01. Consider these safety critical systems in the scope of future technologies and frameworks developed. If the appropriate technology is there, it will be adopted.

02. Recognise that what safety entails is always evolving, and will in fact affect our views on system dependability and security.

03. Standards and guidelines already exist to increase systems’ security and safety. Applying best practices wherever applicable will promote system dependability.
Thank You!

@heidykhlaaf