Runtime Monitoring for the Diagnosis and Recovery of Complex Physical Systems

G. Pace, C. Colombo, K. Vella, G. Valentino (University of Malta, Msida, Malta)

G. De Cataldo, A. Franco (Istituto Nazionale di Fisica Nucleare, Bari, Italy)

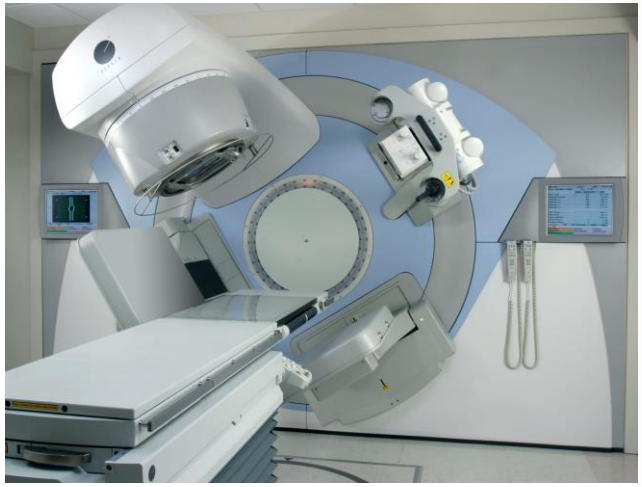
ATTRACT TWD Symposium, Barcelona, Spain, 1st July 2016

Introduction

- Reliability and fault tolerance in the operation of critical systems is **imperative** to avoid financial setbacks or even loss of lives
- Software engineers try to test their software as much as possible… but very difficult to test huge software products thoroughly to ensure correct functionality.
- Such a system does not work in a vacuum, but in an environment for which it is impossible to simulate **all possible variations**.

Murphy's Law:

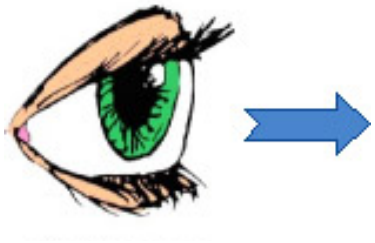
- **Therac-25:** a radiation therapy machine involved in 6 accidents in the 1980s.
- Patients given massive overdoses of radiation due to failure of software interlocks to protect against high-power beams.
- A 1-byte counter in a routine frequently overflowed, and if the operator provided an input at that moment, the interlock would fail.

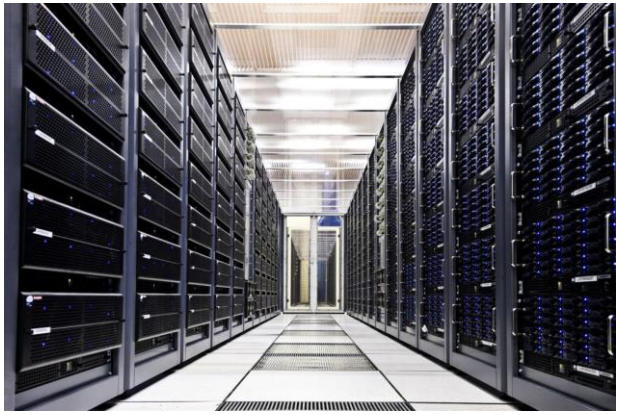


Typical software analysis

- Many automated tools in or outside Integrated Development Environment (IDE) for code coverage, profiling, benchmarking..
- Done at development phase, when bugs are cheaper to fix.
- **Static testing:** code reviews, walkthroughs, inspections.
- **Dynamic testing:** executing code for a given set of test cases.

What is Runtime Verification?

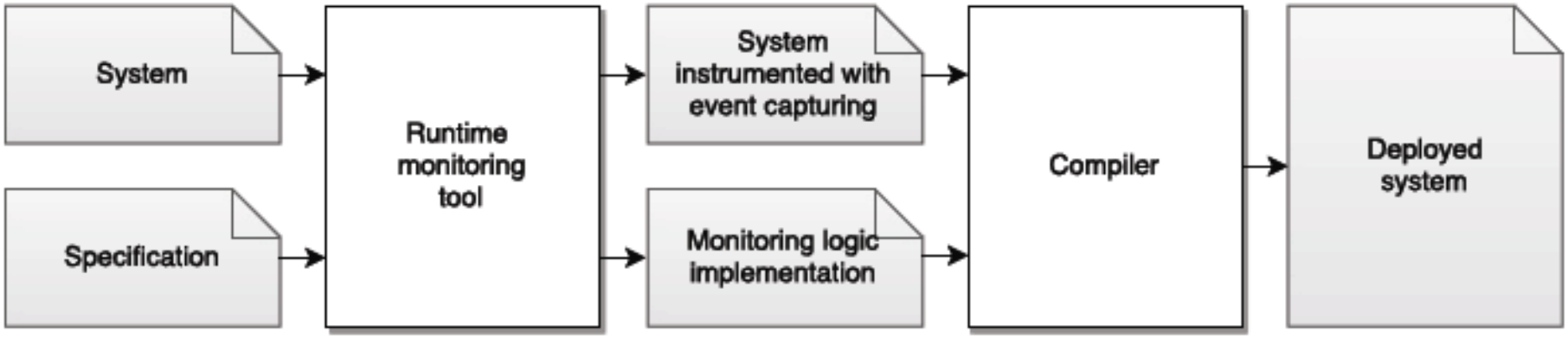




- The extraction of information from a **running** system, to **detect and react** to observed behaviours violating certain properties.

- Rather than **inlining** property checks via e.g. assertions, which **interweaves** the program with the specification of the properties to be satisfied, the two are **separated**.

How does Runtime Verification work?



1. Runtime monitoring tool: modifies the system code to capture points of interest during execution.

2. The specification is automatically converted into a monitor, which checks that the system behaviour does not violate the specification.

How does Runtime Verification work?

- Separation of the system and specification aspects allows for:
  - Different teams working on the different aspects
  - The use of the same specification across different versions, instances or systems
  - Avoids possibility to introduce new bugs as complexity of inline checks increases
  - E.g. *isGasLeak* variable should not have been true for more than 1 minute in the last 30 minutes just before *induceSpark* function is called..

Detector Control System: A use case for RV

- The ALICE High Momentum Particle Identification (HMPID) detector and ALICE-LHC interface control systems are based on the WinCC OA platform.

- Both are critical systems:
  - issues with the detector (e.g. pressure, cooling, HV trips) could result in downtime or even damage
  - issues with ALICE-LHC interface (e.g. DIP server status, disk space, manager status) could result errors in the production of parameter files for the physics runs.

- WinCC OA provides an event-driven platform for controlling and monitoring distributed physical devices and software services.

- These systems are modelled using state machines (SMI++) and the human end-user interacts with them through a GUI.

Detector Control System: A use case for RV

- In both cases, some degree of automatic problem diagnosis already exists through raising of alarms which are visible in the GUI.

- However, typically the recovery from single or multiple failures needs to rely on some form of human intervention, which costs time.

- Runtime monitoring is well-suited to ensuring, using the same specification input, that:

  - the distributed WinCC system does not enter into any critical conditions that could be harmful for the detector or accelerator.
  - the recovery from conditions which occur from time to time is performed automatically.

Our expertise in RV and DCS

- The Semantics and Verification Research Group (SVRG) at the University of Malta has been active in RV since 2005.
- RV applications that we worked on:
  - Industrial financial transactions
  - Telescope signal processing
  - Computer vision (airport security)
- Already involved in various European projects e.g. Open Payments Ecosystem (Horizon 2020), ARVI COST Action.
- Many years of experience in the design, implementation and operation of detector control systems (HMPID & LHC_IF)