Software Patterns for Fault Injection in CPS Engineering

Nicolas Navet [*], Ivan Cibrario Bertolotti [**], *Tingting Hu* [*]

[**] National Research Council of Italy – IEIIT, Torino, Italy
[*] University of Luxembourg – FSTC, Esch-sur-Alzette, Luxembourg

Sept. 12 – 15, 2017, Limassol, Cyprus

Motivation

Model-Driven Engineering and Domain-Specific Languages are two key technologies to meet the software productivity challenge and develop trustworthy systems, especially for CPS.

- CPS typically subject to dependability constraints
- Fault injection is an effective technique to assess the dependability, either implemented in hardware (HIFI) or software (SWIFI)

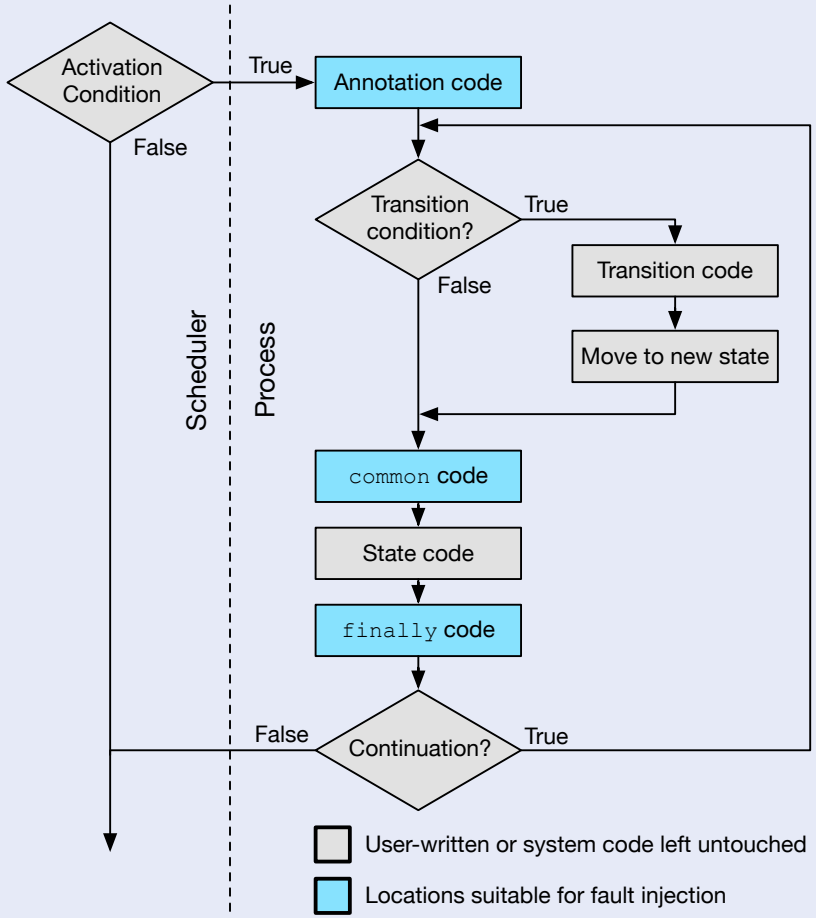
A central challenge in MDE is to make it possible to seamlessly integrate the verification activity within the design flow and to fully, or partially automate it.

CPAL

CPAL sample

```
processdef P(params) {
  common {
    code
  }

  state Warning {
    code
  }
  on (cond) {code} to Alarm_Mode;
  after (time) if (cond) to Normal_Mode;

  finally {
    code
  }
}

process P: inst[period,offset][cond](args);

@cpal:time:inst{
    annotation code
}
```

Elementary execution step



Fault Injection Patterns

Software patterns

Capture structures, ideas, or key techniques known to expert practitioners, and ultimately solve recurring problems.

Fault category

- Global state
- Activation arguments
- Local instance variables
- Control flow disruption

Injection patterns

- External fault injector(s)
- Fault injection as pre/post conditions
- Annotation-based fault injection

Fault Injection Patterns

Software patterns

Capture structures, ideas, or key techniques known to expert practitioners, and ultimately solve recurring problems.

	Global state	Act. args.	Local vars.	Control flow
External process(es)	✓			✓
Pre/post conditions	✓	✓	✓	
Annotation-based	✓			✓

Key feature: automated generation of fault injection code based on code transformation.

Thank you for your attention