Introduction for second half of course

Paul Jackson

School of Informatics
University of Edinburgh

Formal Verification
Spring 2017

Topics for rest of course

Focus mostly on Software FV

- SPARK language and toolkit

  Example of a WP (Weakest Precondition) based approach
  - Overview of language, focussing on verification aspects
  - Labs
  - Tool architecture – use of the Why3 intermediate language
  - Underlying maths and algorithms
    - operational semantics
    - weakest precondition calculation,
    - verification condition generation
  - Provers (SMT solvers, interactive theorem provers, FOL automatic theorem provers)
  - Methodology (e.g. static vs. dynamic assertion checking)
  - Other tools using WP (e.g. Dafny, Frama-C, Leon)

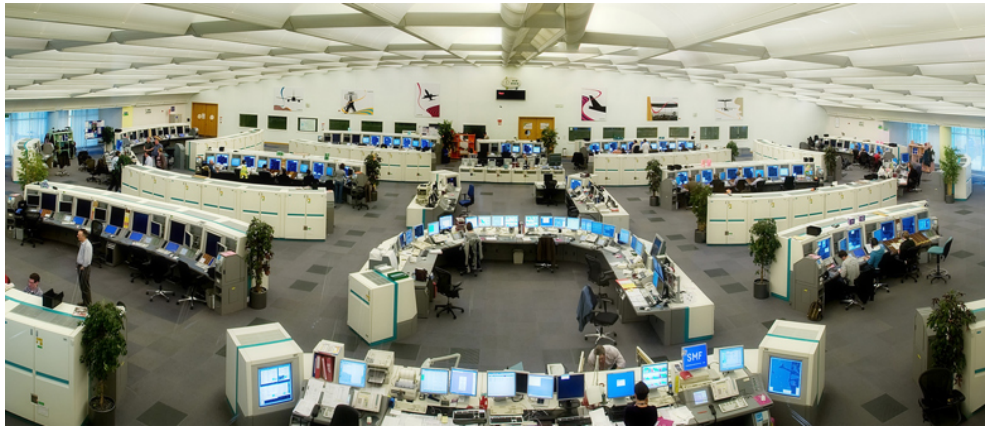
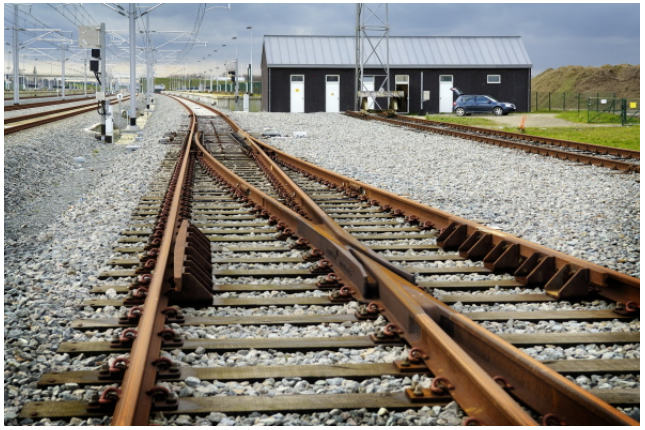
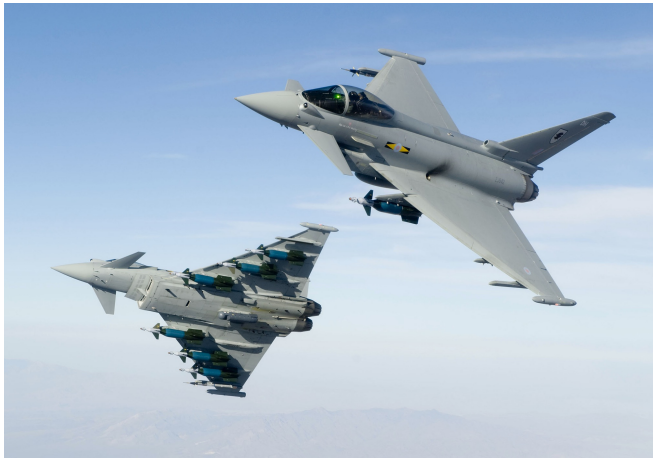
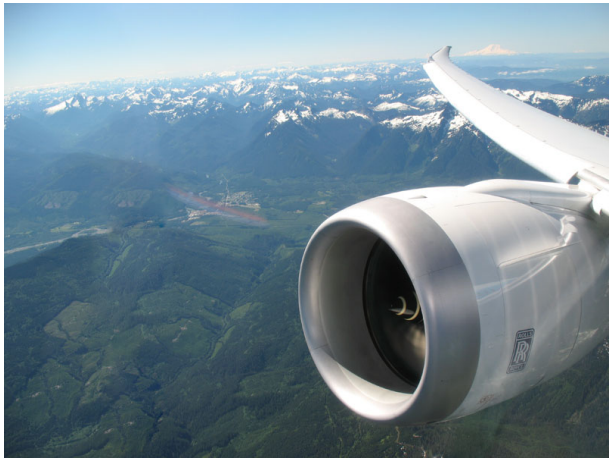
Topics continued

- SAT & SMT algorithms and technology
- CBMC – bounded model checking for C
- Other FV techniques (time permitting)
  - Abstract interpretation, predicate abstraction, interpolation, ...
- Bigger picture
  - Current take-up of FV by industry
  - Research challenges

SPARK language overview

- Subset of Ada
- Adds in features for verification
- Designed for high-integrity (safety/security/mission critical) applications
  - Syntactical features make mistakes harder
  - Strong typing
  - No undefined behaviours

SPARK application examples


Pictures provided by Altran UK

SPARK tools

- Commercially developed & supported (Altran and Adacore)
- Based on free software (gcc, Why3, CVC4)
- GPL licensed (mostly . . .)
- Include
  - GNAT gcc-based compiler
  - GPS IDE
  - Plug-in for Eclipse IDE
  - GNATProve formal verification tool
- FV support includes:
  - Flow analysis
  - Ensuring freedom from run-time errors
  - Property checking
  - Functional verification