Seminari Cuc

Current trends in industry applications of software verification tools

Fabian Romero

IRIT, UT3 Paul Sabatier

Introduction

Large projects

- compcert
- eChronos RTOS
- Little Bird
- Project Everest

Tools

- Coq
- Isabelle
- F Star
- ACL2
- GHilbert
- Hilbert II
- HOL Light
- Metamath
- ProofPower

The line of code and the mythical man month

How is LOC still a thing?

industry	fault density Error for Kloc
Automotive	3
Aviation	1
Shuttle	0.1
Traditional	200
Agile	22

http://leanagilepartners.com/publications.html

Software Reliability Models

- Many Models exist (ca. 40 published)
- Different Assumptions of Models:
- Definition of testing process
- Finite or infinite number of failures?
- No faults introduced during debugging?
- Distribution of data (Poisson, Binomial)?
- Data requirements? (inter-failure data, failure count data)
- Assessing the goodness-of-fit
- Kolmogorow-Smirnov, Chi-Square
- Usually, Reliability Models assume reliability growth.

Engineering pitfalls

- Software is not a good that can be mass produced
- nor dramatically pipelined
- software errors are not ”failures” with MTBF and other engineering measures
- software errors are deterministic

Cost

Non scientific or statistical data

from: mobility network logistics

The reduction of errors across the industry has a exponential cost.
Roughly cutting by half the rate of errors doubles the cost
Formal verification is in principle in many cases, and in practice in some cases, able to produce zero-defects software and has a ”fixed” 20x cost than regular enterprise techniques.

Initials Roadblocks

- Formal verification is not the same than functional verification.
- Specification is the main problem for implementation, verification is not the hard problem
- Writing specification for legacy systems is considerably harder Systems that haven't been designed with verification in mind might not provide an appropriate component structure. Even if they obey principles such as information hiding and encapsulation, their components might not be of the right granularity or might have too many interdependencies.

Handling Complexity

- Abstraction .
- Descomposition
- Testing
- Integration testing
- Automatic testing