TATA CONSULTANCY SERVICES
Experience certainty.

Software Verification

Trends and Challenges

December 6, 2016

Talk Aims

Introduction
- Technologies, techniques

Current state
- Industry code
- Large applications

Reality check

Cost of Software Bugs

Company	Year	What & why	Source
Maquet	2011	Anesthesia systems	Fda.gov
BMW	2012	7-series vehicles – door latching problem	www.nconsumer.org
Volvo	2012	S80 vehicles – possible engine stall	www.nconsumer.org
Knight Capital	2012	Bought and sold shares at a loss $440m loss	New scientist
Amazon	2014	Items sold at 1p	computerworlduk
Lockheed Martin	2015	F35 detects targets incorrectly	Fox news
Nissan	2015	Airbags do not inflate	Computer world UK

One of the top three causes of medical devices recalls (Stericycle Expert Soln)

The Future - Robots Everywhere

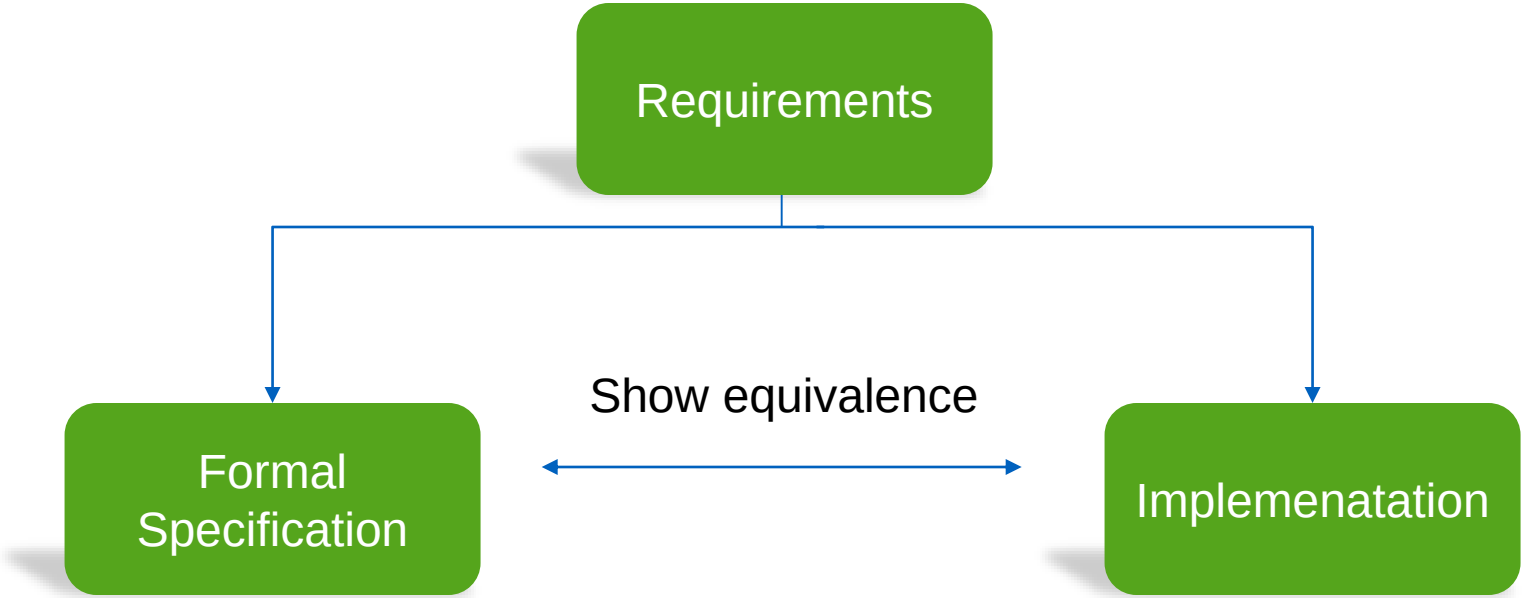
- Autonomous vehicles
- Complex surgery
- Rescue operations
- Complex decision making

Correct Software

Terminologies

- Proving
  - Software meets requirements
- Testing
  - Software runs correctly for given inputs
- Verification
  - Software satisfies certain properties

Proving



- Undecidable in the general case, intractable in most cases
- Large systems – no complete requirements
- Creating formal specification expensive

Testing

- Guarantees (almost) nothing!!
- Most practical
- Validating runs expensive
- Hard to find certain bugs
  - Concurrency, security …

Verification

- Guarantees w.r.t properties (mostly)
- Needs a good list of properties - impractical
- Scalability and precision

Properties

Platform/generic properties

- No crashes due to
  - Division by zero
  - Overflow or underflow
- No hanging
  - Deadlock, livelocks

Domain Properties

- Stop within t secs of braking
- A debit for every credit
- Don't sell at a loss

Nonnegative i = *, j = *

```
If ( j < i )        j' = i
else                j' = j + 1
i/j';   // divide by zero?
```

```
cr(ac, am)          db(ac, am)
  b = getbal(ac)      b = getbal(ac)
  b = b + am          b = b - am
  setbal(ac,b)        setbal(ac,b)
```

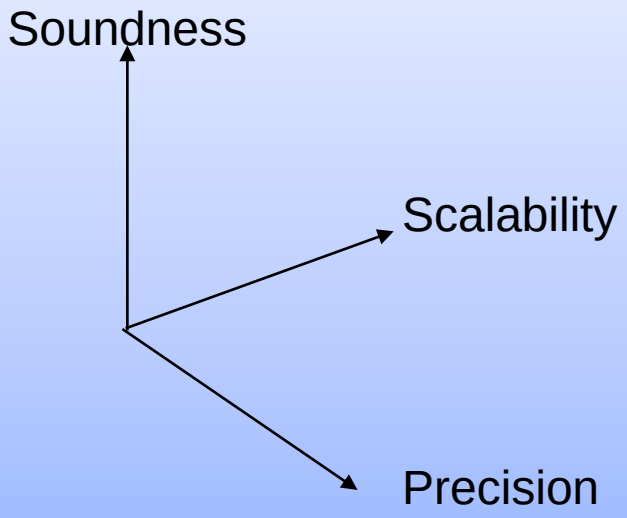
```
xfer(ac1, ac2, am)
  cr(ac2, am)
  || db(ac1, am)
```

Soundness, Precision, Scalability

Soundness
- OK reports are correct

Precision
- Error reports may or may not be errors
- False positives

Scalable
- Can analyze large systems



Technology	Attributes
Static Analysis	Sound, Scalable, Imprecise
Model Checkers	Sound, Precise, Not-scalable
Heuristics based analysis	Unsound, Precise, Scalable

Technologies

Overview

Static Analysis

- Old
- Very abstract
- Too many false alarms

SAT, SMT

- Precise
- Recent advances

Static Analysis & Abstract Interpretation

- Analyse without executing
  - Track properties
- Standard properties
  - Zero division, array index
- Abstract representation of program
- Imprecise
  - Need to know maths
- Abstract interpretation
  - Range, difference, polyhedral

```
nonnegative i = *, j = *
[ j & i can be zero]
If ( j < i )      j = i
else              j = j + 1
[j can be zero]
i/j;   // divide by zero?
```

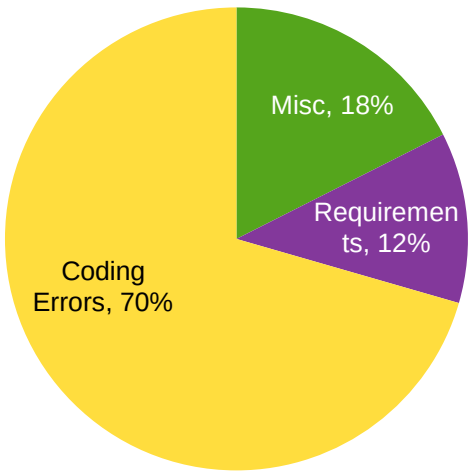
Value

Defect analysis

- >30% of defects

Case studies

- office automation system
  - several defects in production code
  - $1m per year saving
- vehicle infotainment system
  - deep bugs detected
  - 60% effort saving in review time



Analysis of application defects found during product testing

Challenges

Application	Size	Key Characteristics	Warnings
Infotainment	2MLOC(1 task)	Large, large arrays(512), loops(unknown bounds)	77 (ZD)
Smart card component	7K	Loops with large bounds and unknown bounds	55 (ZD)
Auto ECU	6K	Complex control algorithms	128 (AIOB), 43 (ZD)

```
Int a[512];
j = random() * 2;

for ( ; j < 512; j += 2)
        a[ j+1 ];
```

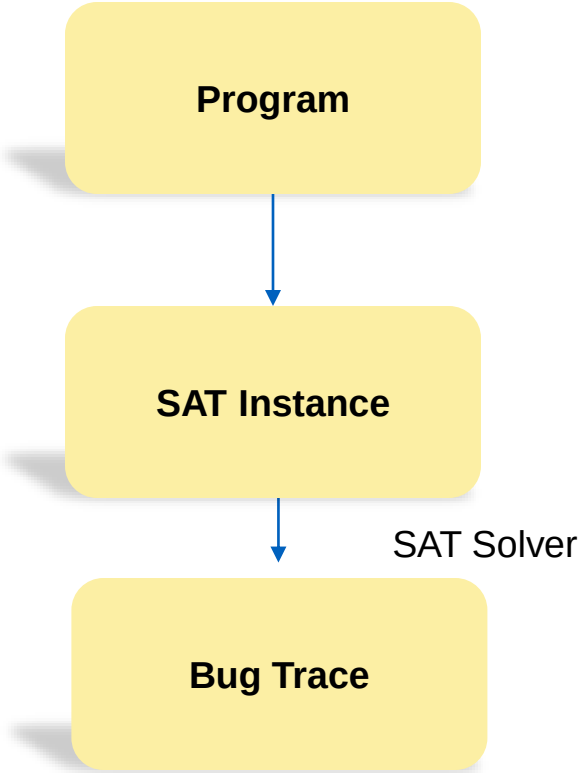
```
int secs[12] = { … }
t = *
m = 0

while(t  > secs[m])
    t  = t – secs[m]
    m = m + 1
```

Satisfiability Checking

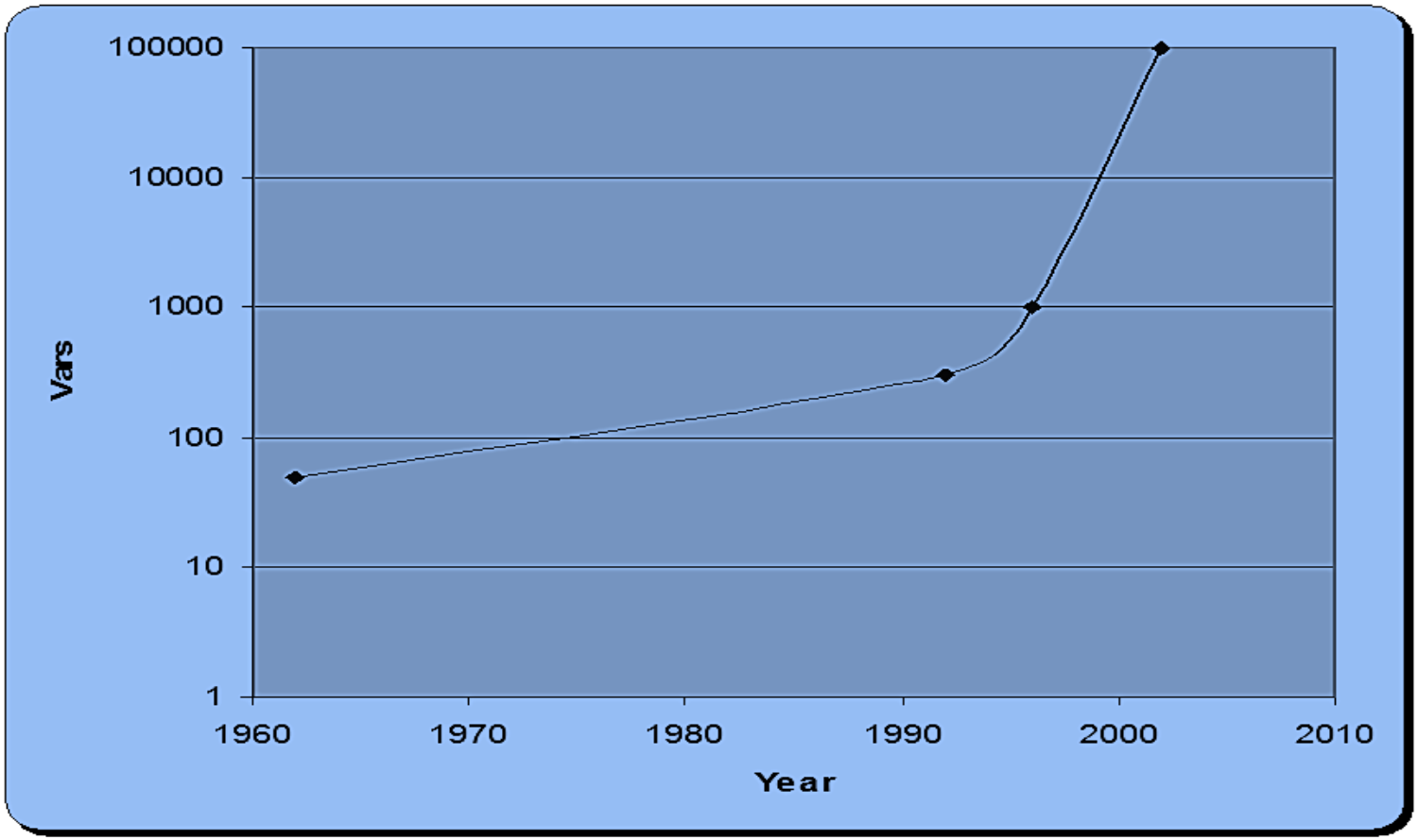
- SAT solving
  - Checking satisfiability of propositional formulas

    $$(rain \rightarrow wet) \wedge (rain \wedge \neg wet)$$

  - NP-complete (Cook)
- Programs – SAT
  - Finite programs
  - a/x ; x == 0 satisfiable?



SAT Solver Performance



Graph thanks to Daniel Kroening

Applications of SAT Solving

- Planning
- Optimizations
  - Knapsack,
- Combinatorial problems
  - Sudoku
- Test pattern generation

CBMC

C Bounded Model Checker

- sound, very precise, low scalability

BMC

- unroll loops finite number of times
- very successful in h/w
- appropriate for embedded systems
- small model hypothesis

Free download

- http://www.cprover.org

Another Problem Case

```
int sq1 ( int y )
    int z, x
     z = y, y = x, x = z
     return x*x
```

```
int sq2 ( int y )
    return y*y
```

y = *

sq1(y) == sq2(y) ?

SMT Solvers

- Theories work better
  - Bit arithmetic
  - Arrays
  - Strings
  - Uninterpreted functions
- Limited scope
- Combine Theories with SAT
  - Satisfiability Modulo Theories (SMT)

$$z = y \wedge y1 = x \wedge x1 = z$$
$$\wedge$$
$$ret1 = sq(x1)$$
$$\wedge$$
$$ret2 = sq(y)$$
$$\wedge$$
$$ret1 \neq ret2$$

SAT v/s SMT - Performance

```
int sq1 ( int y )
        int z, x
        z = y, y = x, x = z
        return x*x
```

```
int sq2 ( int y )
        return y*y
```

---

y = *

sq1(y) == sq2(y) ?

SAT takes twice as much time as SMT

Loops

```
int secs[12] = { … }
t = *
m = 0

 while(t  > secs[m])
    t  = t – secs[m] //err?
    m = m + 1
```

```
while(n != 0)
   lock();
     n = *
    if (n != 0 ) unlock()

unlock(); //err?
```

```
Int a[512];
j = random() * 2;
for ( ; j < 512; j += 2)
            a[ j+1 ];//err?
```

SMT and SAT fail - Unknown bounds, Large bounds

Loop Abstraction and Induction

```
while(n != 0)
{
    lock();    //err?
    n = *
    if (n != 0) {
       unlock()  //err?
    }
 }
 unlock(); //err?
```

```
while(n != 0)
{
  lock();
  n = *
  if (n != 0 ) {
     unlock()
  }
  # unlock xor n == 0
  if (n == 0)   unlock(); //err
}

if (n == 0)   unlock(); //err
```

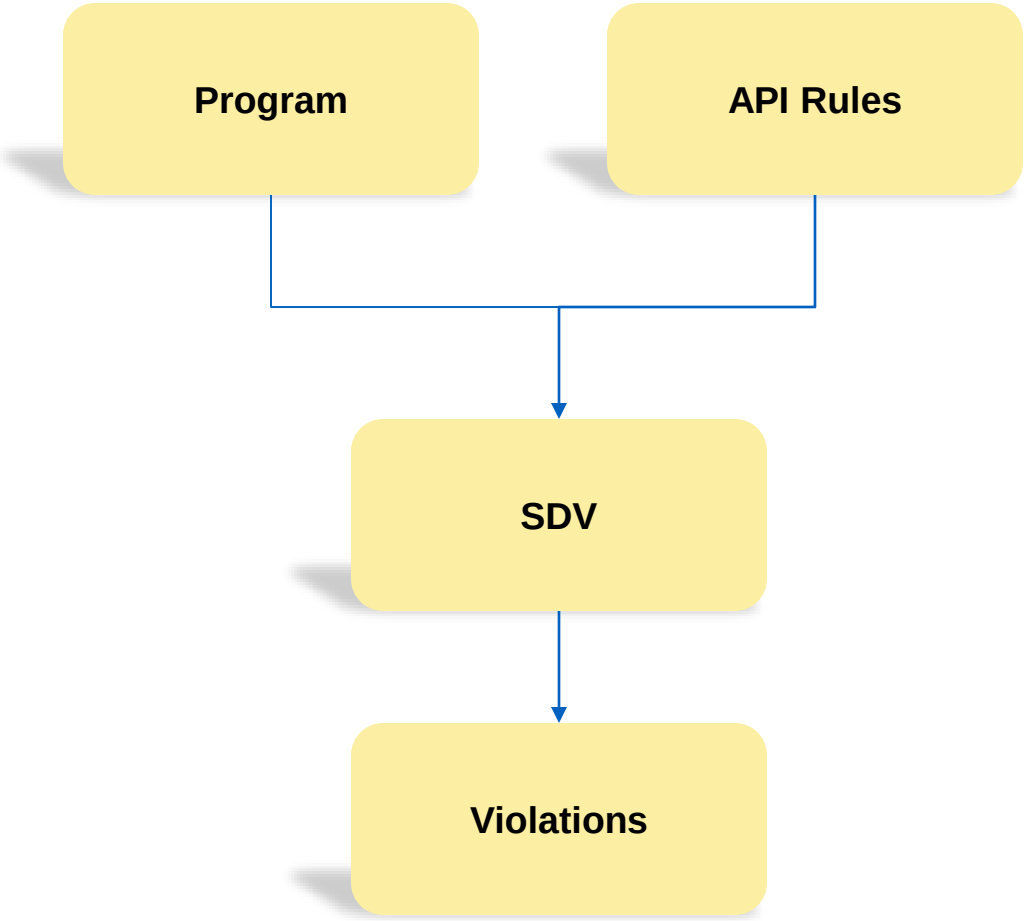
Abstractions on Industry Code

Embedded Application	KLOC	TCS ECA alarms	TCS ECA + LABMC alarms	% precision improvement	Avg. elimination time per alarm (mins.)	TECA + LABMC execution time
A1 – Protocol stack	8	94	29	69.15	0.15	13 min.
A2 – Office automation	4.6	196	92	53.06	0.30	59 min.
A3 – Car S/W	34	346	251	27.46	0.29	1 hour 40 min.
A4 – Battery controller	60	189	62	67.20	0.37	1 hour 9 min.
A5 – CAN driver	18.3	226	66	70.80	0.21	47 min.
A6 – Vehicle navigation system	184	422	145	65.64	1.41	9 hours 55 min
A7 - Vehicle S/W	171.4	309	144	53.40	1.87	9 hours 37 min.

Applications

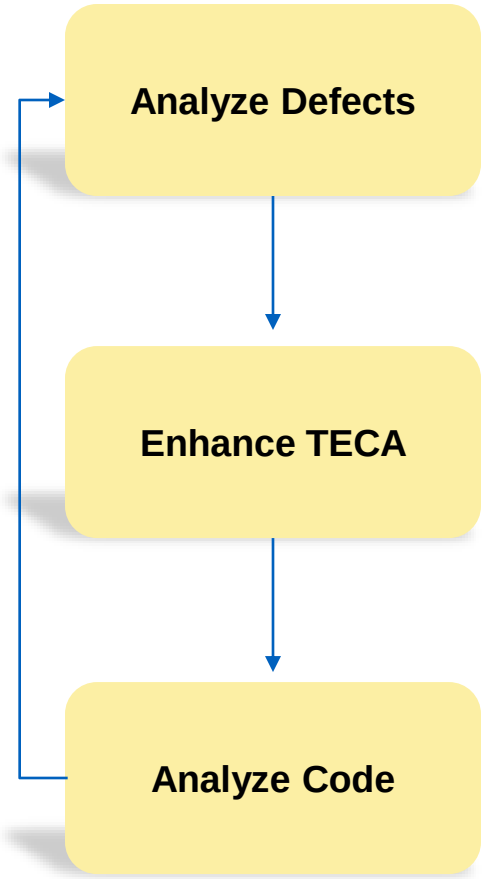
Driver Verification

- Microsoft
  - Slam project
- Static Driver Verifier
  - Automates CEGAR
- Windows 7 drivers
  - 270 bugs (tested code)
  - CACM Jul ‘11
- Similarly for earlier versions



Towards Zero Defects

- TCS
  - TCS Embedded Code Analyzer (TECA)
- Auto Infotainment System
  - Static analysis
  - 20+ defect categories
- 10M lines of code
  - Several defects
  - 60% reduction in review time



Reality Check

- Current state
  - Verification of MLOC
  - Sequential code
- Modern Cars
  - Billion LOC
  - More than 100 ECUs
- Sophisticated algorithms
  - Image processing

TATA CONSULTANCY SERVICES
Experience certainty.

TATA

Thank You

IT Services
Business Solutions
Consulting