Quality Check Tools for Software Developers

Francesco Giacomini (INFN/CNAF)

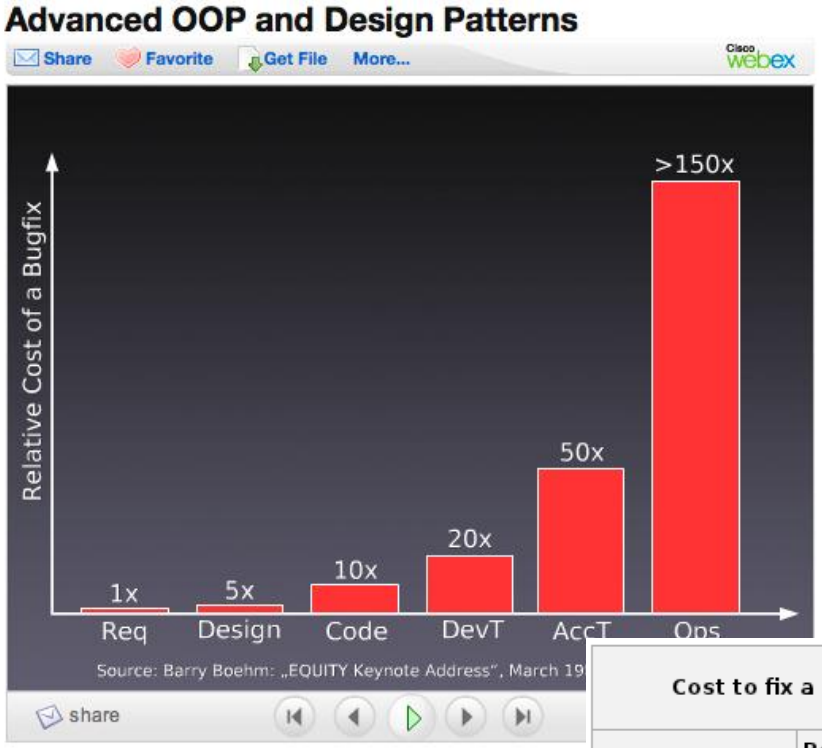
SuperB Computing Workshop
Ferrara – 4-7 July 2011

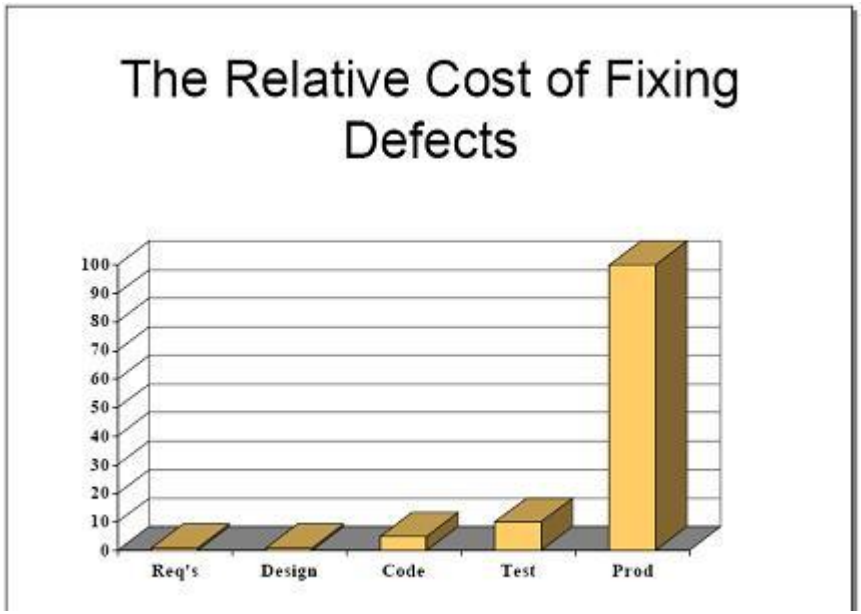
Outline

- The problem
  - Software bugs
- How to mitigate it
  - Static analysis tools
- Examples

The problem: economical view

- Investigating and fixing bugs is expensive





Cost to fix a defect		Time detected				
		Requirements	Architecture	Construction	System test	Post-release
Time introduced	Requirements	1×	3×	5-10×	10×	10-100×
	Architecture	-	1×	10×	15×	25-100×
	Construction	-	-	1×	10×	10-25×

The problem: sociological view

- Developing code is fun
- Debugging and fixing code is frustrating
- Debugging other people’s code is even more frustrating

How to address the problem

- There is no silver bullet
  - Many approaches and techniques exist
- Goal: identify software defects as early as possible
- Static analysis tools
  - “Enhanced compiler”
  - Early identification of possible bugs or potential occasions of bugs
  - Enforcement of coding standards
  - Dataflow analysis without execution

Examples

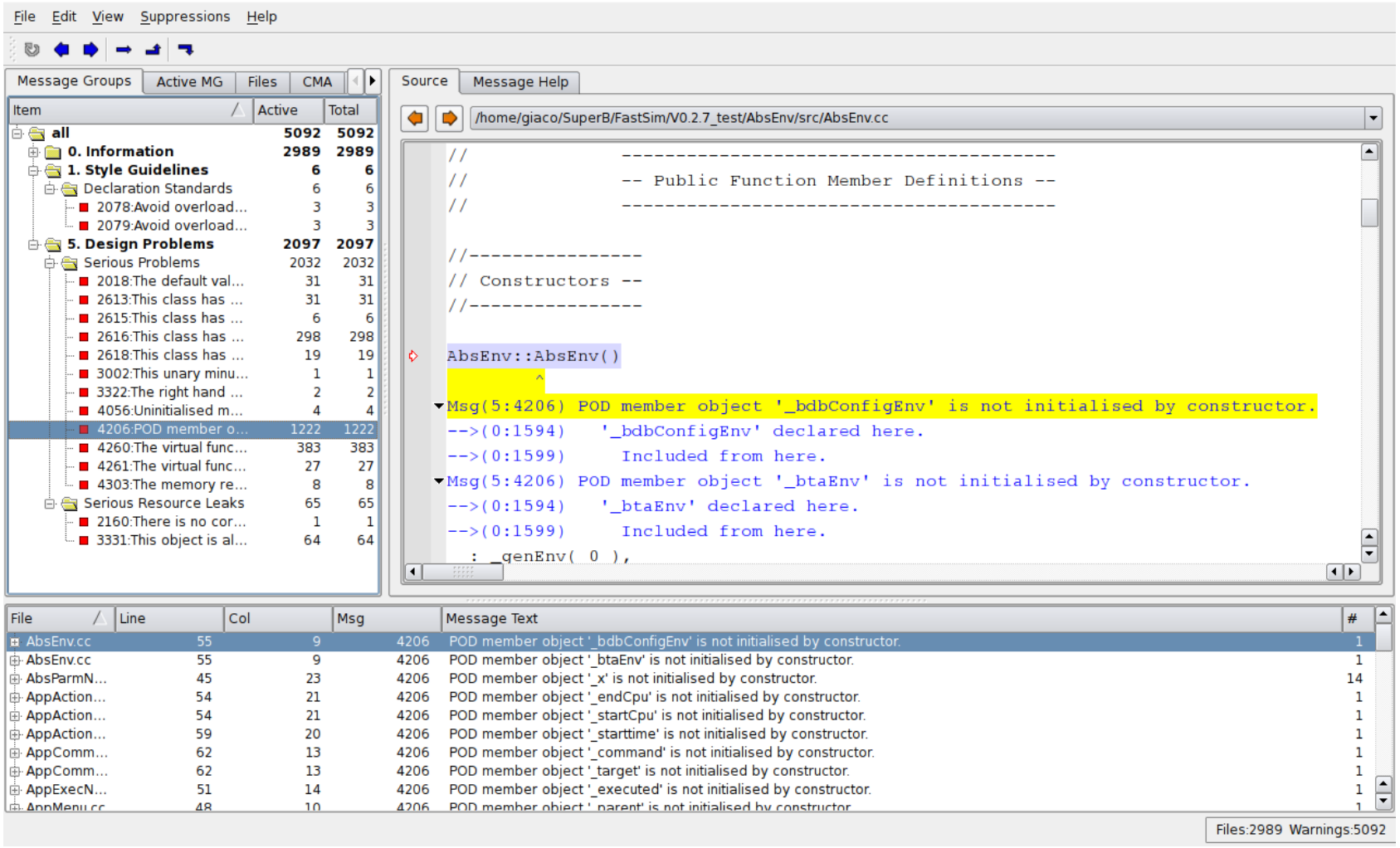
- Commercial Tool 1
- CppCheck
- CCCC

- Applied to FastSim V0.2.7_test

Commercial Tool 1

- Checks compliance with published C++ coding standards
  - QA·HIC++
  - QA·MISRA-C++
  - QA·JSF++

Commercial Tool 1 GUI



Commercial Tool 1 Findings /1
(only *critical* issues)

```
if(_nHidden>0) fitpar->addChiSquare(0,-_nHidden) ;
                                     ^

Msg(5:3002) This unary minus operator is being
applied to an unsigned type.


binTime.upper = (scalerTime >> 32) & (0xffffffffUL);
                           ^

Msg(5:3322) The right hand side operand of the shift
operator is too large.
```

Commercial Tool 1 Findings /2

```
AbsEnv::AbsEnv()
        ^
Msg(5:4206) POD member object '_bdbConfigEnv' is not
initialised by constructor.
```

(many of them)

Many mismatches between copy constructor, operator= and destructor (all of them should be there or none of them)

Commercial Tool 1 Findings /3

```
delete [] _thickness;
for(unsigned i=0; i < _nelems; i++){
  _thickness[i] = new d_Double [_nsegs];
  ^
Msg(5:4303) The memory referred to by '_thickness'
has been deallocated.


delete _reftraj;
if(_reftraj != _ptraj)
   ^
Msg(5:4303) The memory referred to by '_reftraj' has
been deallocated.
```

Commercial Tool 1 Findings /4

```
if (_menuFileName) {
  delete _menuFileName;
         ^
Msg(5:3331) This object is also used as pointer to
array.
}
_menuFileName = new char[strlen(filename)+1];
```

```
pointer= new Char_t;
^
Msg(5:3331) This object is also used as pointer to
array.
```

False positive

Commercial Tool 1 Findings /5 (in Root)

```
template <class Element>
TMatrixT<Element> operator&&(...);
                  ^
Msg(1:2078) Avoid overloading operator 'and' (&&).


void *operator new(size_t l) { ... }
      ^
Msg(5:2160) There is no corresponding operator
delete for this operator new.
```

(False positive, I hope)

CppCheck

- An open-source static analysis tool for C/C++ code
  - http://cppcheck.sourceforge.net/
- *Features*
  - *Out of bounds checking*
  - *Check the code for each class*
  - *Checking exception safety*
  - *Memory leaks checking*
  - *Warn if obsolete functions are used*
  - *Check for invalid usage of STL*
  - *Check for uninitialized variables and unused functions*
- Goal is zero false positives

CppCheck Findings

- (warning) Member variable 'ThreeDCoord::_uvw' is not initialised in the constructor.
- (style) 'TrkSvtSLayoutT::operator=' should return 'TrkSvtSLayoutT &‘
- (warning) Member variable 'IfdKey::_hashVal' is not assigned a value in 'IfdKey::operator=‘
- (style) The class 'KalmanCalculator' does not have a constructor
- (performance) Function parameter 'formalCmd' should be passed by reference.
- (performance) Prefer prefix ++/-- operators for non-primitive types.
- (style) Variable 'rval' is assigned a value that is never used.
- (warning) Possible leak in public function. The pointer '_theTargetName' is not deallocated before it is allocated.
- (information) The scope of the variable 'index' can be reduced
- …

Further step: control complexity

- Keep code complexity under control
- More complexity → more testing
- Metrics exist to measure complexity at different levels
  - Function, class, file, program
- Functionality available in several tools
  - Commercial and open-source

CCCC Findings
(on Framework)

- C and C++ Code Counter
  - http://cccc.sourceforge.net/

Metric	Tag	Overall	Per Module
Number of modules	NOM	76	
Lines of Code	LOC	5579	73.408
McCabe's Cyclomatic Number	MVG	929	12.224
Lines of Comment	COM	2811	36.987
LOC/COM	L_C	1.985	
MVG/COM	M_C	0.330	
Information Flow measure (inclusive)	IF4	628	8.263
Information Flow measure (visible)	IF4v	628	8.263
Information Flow measure (concrete)	IF4c	0	0.000
Lines of Code rejected by parser	REJ	419	

Conclusions

- The later bugs are discovered, the more it costs to fix them
- Static analysis tools exist to identify real or possible bugs very early in the development cycle
- Let’s use them