Building Trust into OO Components using a Genetic Analogy

Yves Le Traon
Yves.Le_Traon@irisa.fr

IRISA

Summary

- The problem: trustable components
- Component qualification with mutation analysis
- Genetic algorithms for test enhancement
- A new model: bacteriological algorithms for test enhancement
- A mixed-approach and tuning of the parameters
- Conclusion

IRISA

Testing for trust

Object oriented paradigm

Reuse

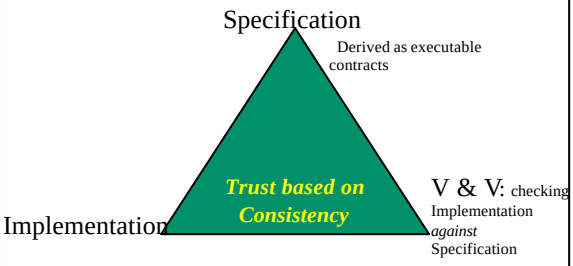
Safety

Trust

the better you test, the more trustable your component is

IRISA

Testing for trust



IRISA

Testing for trust

- We use a *mutation analysis*
  - we inject simple errors in the class
    - we create *mutants*
  - we write *tests* to detect the mutants
    - the tests *kill* the mutants
  - two ways to detect mutants
    - by comparing behaviors
    - with an oracle function: contracts

Component qualification with mutation analysis

Mutation operators
Case study
The problem of automating the test enhancement process

IRISA

Limited Mutation Analysis

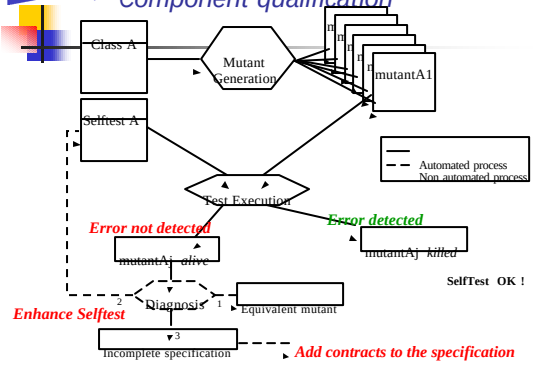
```
put (x : INTEGER) is
    -- put x in the set
  require  not full: not full
  do
1   if not has (x) then
2     count := count + 1
3     structure.put (x, count)
    end -- if
  ensure
    has: has (x)
    not empty: not empty
end -- put
```

- 1

Remove-inst

IRISA

Component qualification



IRISA

Component qualification

Mutation operators

Type	Description
EHF	Exception Handling Fault
AOR	Arithmetic Operator Replacement
LOR	Logical Operator Replacement
ROR	Relational Operator Replacement
NOR	No Operation Replacement
VCP	Variable and Constant Perturbation
MCR	Methods Call Replacement
RFI	Referencing Fault Insertion

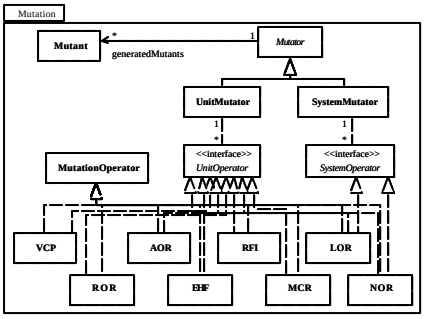
IRISA

Mutation for Unit and System testing

- System testing
  - Combinatory explosion of the #mutants
  - Determination of equivalent mutant unrealistic
- A subset of operators must be chosen to deal with these constraints
- A strategy for injecting faults ?
- Specific operators ?

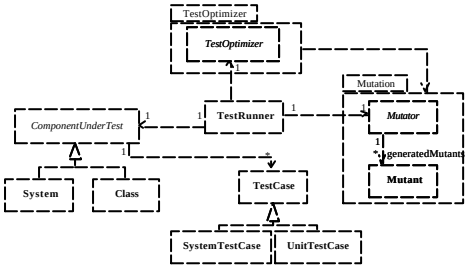
IRISA

Mutation for Unit and System testing



IRISA

Mutation for Unit and System testing



IRISA

Component qualification

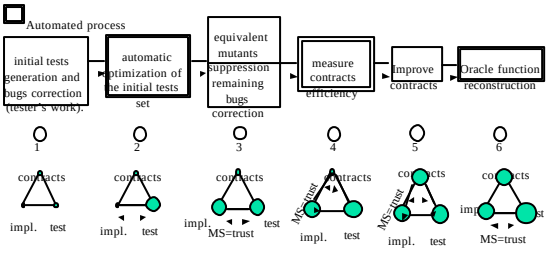
→ Trust estimating : Score of mutation analysis

$$MS = \frac{d}{m}$$

- d : Number of killed mutants
- m : Number of generated mutants which are not equivalent

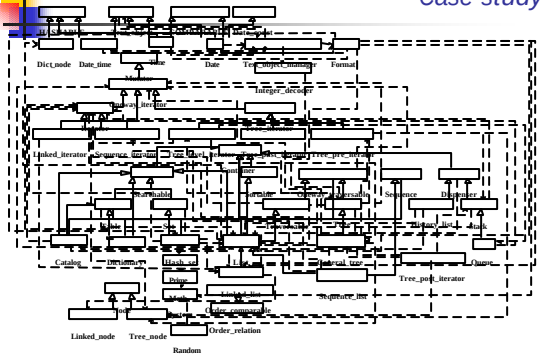
IRISA

Global Process



IRISA

Case study



Genetic algorithms for test enhancement

IRISA

Test enhancement

- Easy to reach a mutation score of 60%
- Costly to improve this score
- Test enhancement ⇨ genetic algorithms

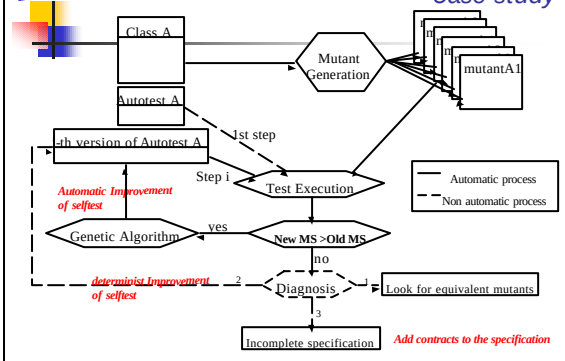
IRISA

Gas for test optimization

- The issue : improving test cases automatically
  - Non linear problem
  - Gas may be adapted

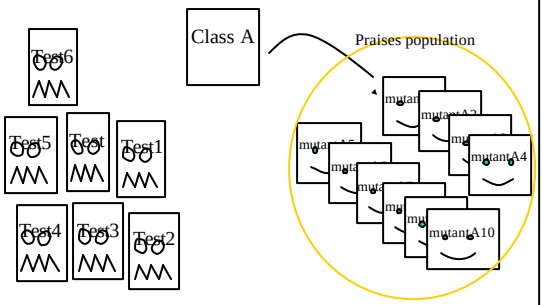
IRISA

Case study



IRISA

Test enhancement



IRISA

Test enhancement

- The analogy
  - Test cases = predators = individuals
  - Mutants = population of prey

IRISA

Test enhancement: Genetic algorithm

- Genes
- Individual
- Operators
  - reproduction
  - crossover
  - mutation
- Fitness function

IRISA

Test enhancement: Genetic algorithm

- Fitness function ⇒ Mutation score
- Individual ⇒ Tests cases
- For unit class testing
  - Gene {Initialization, Function calls}

IRISA

Test enhancement: Genetic algorithm

- Operators
  - Crossover

$ind_1 = \{G_{1\,1}, \ldots G_{1\,i}, G_{1\,i+1}, .. G_{1\,m}\}$ ~~$ind_2 = \{G_{2\,1}, \ldots G_{2\,i}, G_{2\,i+1}, .. G_{2\,m}\}$~~

$ind_3 = \{G_{1\,1}, \ldots G_{1\,i}, G_{2\,i+1}, .. G_{2\,m}\}$ $ind_4 = \{G_{2\,1}, \ldots G_{2\,i}, G_{1\,i+1}, .. G_{1\,m}\}$

  - Mutation for unit testing

$G_1 = [I_1, S_1]$ $G_2 = [I_2, S_2]$

$G_3 = [I_2, S_1]$ $G_4 = [I_1, S_2]$ $G_5 = [I_1, S_1 S_2]$ $G_6 = [I_2, S_2 S_1]$

$G = [I, S] \rightarrow G = [I, S_{mut}]$

$S = (m_1(p_1),\ldots,m_i(p_i),\ldots m_n(p_n)) \rightarrow S_{mut} = (m_1(p_1),\ldots,m_i(p_{imut}),\ldots m_n(p_n))$

IRISA

Test enhancement: Genetic algorithm

- ***Gene modeling for system testing***
  - Must be adapted to the particular system under test
  - In the case of the studied system (a C# parser)
    - If there are x nodes N in the file a gene can be represented as follows

    $G = [N_1,\ldots,N_x]$
  - **Mutation operator for system testing.** *The mutation operator, chooses a gene at random in an individual and replaces a node in that gene by another one:*

    $G = [N_1,\ldots, N_i,\ldots, N_x] \Rightarrow G_{mut} = [N_1,\ldots, N_{imut},\ldots, N_x]$

IRISA

Test enhancement: Genetic algorithm

The global process of a genetic algorithm

- choose an initial population
- calculate the fitness value for each individual

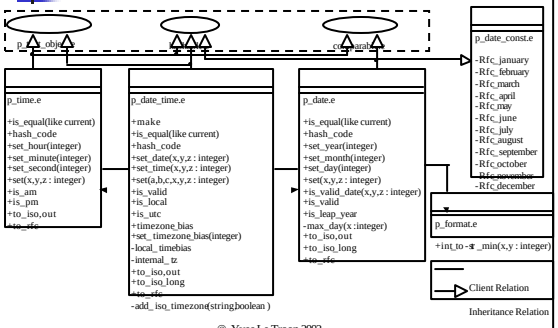
Genetic loop

- reproduction
- crossover
- mutation on one or several individuals

- several stopping criteria : x number of generations, a given fitness value reached …

IRISA

Case study: unit testing



IRISA

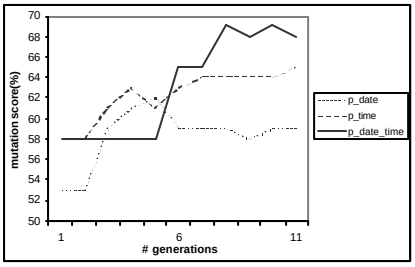
Case study: unit testing

	p_date	p_date_time	p_time
# of generated mutants	673	199	275
mutation score (%)	53	58	58

IRISA

Case study: unit testing

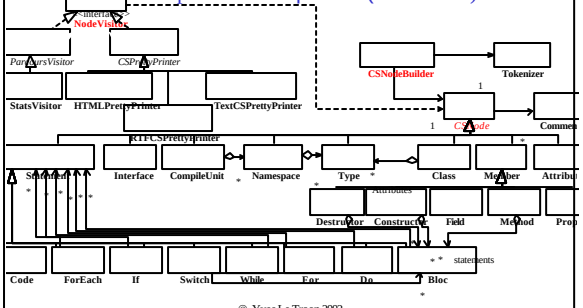
→ Results for unit testing (Mutation rate : 10%)



IRISA

Case study: system testing

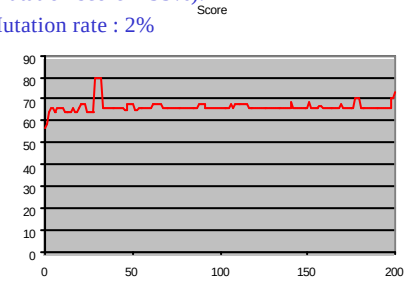
→ a .NET component : C# parser (32 classes)



IRISA

Case study: system testing

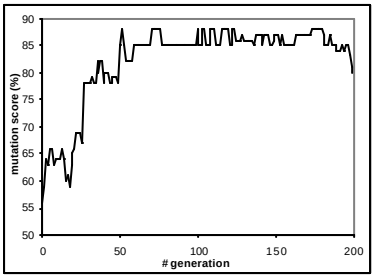
- initial population of 12 individuals of size 4 (mutation score = 55%).
- Mutation rate : 2%



IRISA

Case study: system testing

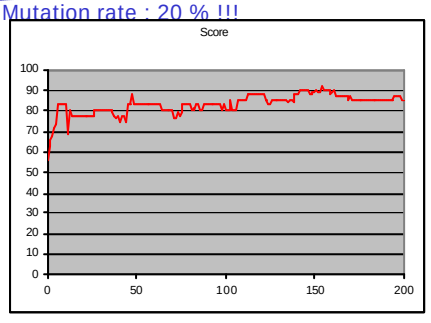
- Mutation rate : 10 %



IRISA

Case study: system testing

- Mutation rate : 20 % !!!



IRISA

Case study: system testing

- The GA have many parameters: difficult tuning
- Results are not stable
- The mutation plays a crucial role
  - Not a « classical » GA
- Some technical reasons for these deceiving results
  - No memorization to guarantee a growth of the MS

A new model: bacteriological algorithms for test enhancement

IRISA

The bacteriological approach

- an adaptive approach
  - Test cases have to be adapted to a given « environment »
  - No cross-over
- A new model taken from a biological analogy
  - The bacteriological approach

IRISA

The bacteriological approach

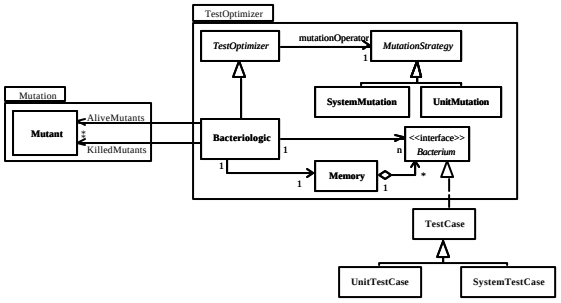
Bacteriological loop

- choose an initial set of individuals
- compute the fitness value for each individual
- memorization of the best individuals
- reproduction
- mutation on one or several individuals
- several stopping criteria : x number of generations, a given fitness value reached …

IRISA

The bacteriological approach

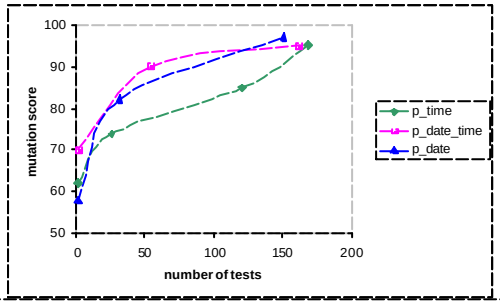
Generic UML model



IRISA

Case study: unit testing

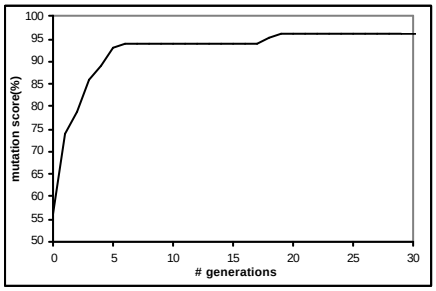
→ Results for unit testing



IRISA

Case study: system testing

→ Initial mutation score = 55%.



IRISA

Comparison

→ Performances

Algorithm	# generation	mutation score (%)	# mutants executed
Genetic	200	85	480000
Bacteriologic	30	96	46375

→ reduced tuning effort: less parameters (size of an individual, selection of individuals for reproduction)

A mixed-approach
Tuning of the parameters

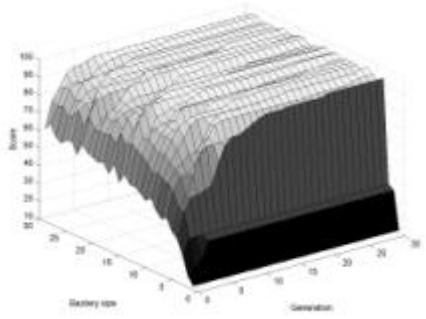
IRISA

Tuning of models and clues for a mixed-approach

- Fix the « size » of a bacterium
  - Study of the C# parser
  - Size of a bacterium = #syntactic nodes

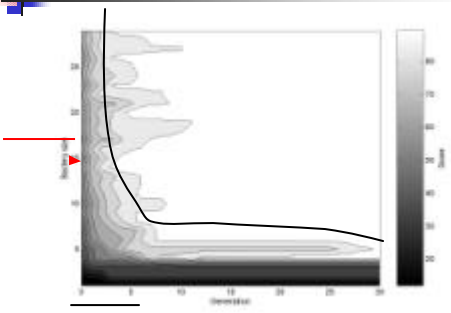
IRISA

Tuning of models and clues for a mixed-approach



IRISA

Tuning of models and clues for a mixed-approach



Top mutation score trend curve

IRISA

Tuning of models and clues for a mixed-approach

Mixed-Approach

→ Loooking for an intermediate solution between 'pure' GAs and bacteriological algorithms.

↳ From no memorization to a systematic memorization

↳ Trade-off between

- number of test cases = number of memorized bacteria
- Convergence speed = number of generations

IRISA

Clues for a mixed-approach

Let B be bacterium and threshold_value be the memorization threshold

if fitness_value(B)> threshold_value **then** memorize B

The category of the algorithm depends on the threshold value:

if threshold_value = 100 **then** "pure " genetic

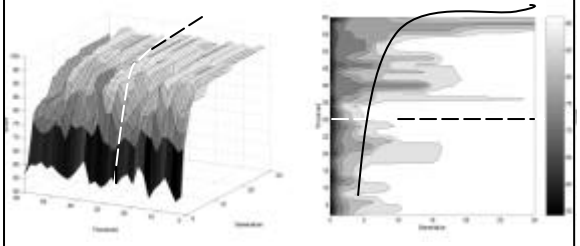
if threshold_value = 0 **then** "pure" bacteriological

if 0 < threshold_value < 100 **then** mixed-approach

IRISA

Clues for a mixed-approach

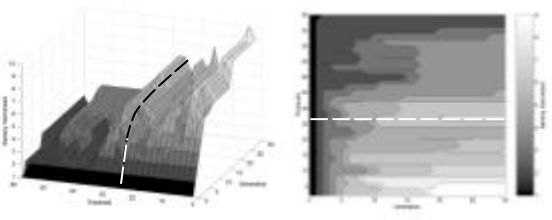
Convergence speed



IRISA

Clues for a mixed-approach

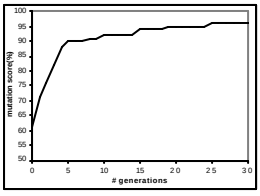
of test cases set = # memorized Bacteria



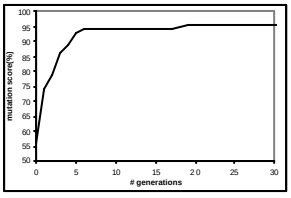
IRISA

Clues for a mixed-approach

A slice for the mixed-approach



Bacteriological approach



Lowest convergence speed for the mixed-approach
Test cases set reduced from 10 to 7
But many parameters must be tuned => bact. still better

IRISA

Conclusion

- Method to build trustable components
- Automated test improvement
  - Gas not adapted
  - BAs better and easier to calibrate
  - Mixed-approach benefit not obvious