Early WCET Prediction using Machine Learning (Work in Progress)

Armelle Bonenfant[1], Denis Claraz[2], Marianne de Michiel[1] and Pascal Sotin[1]

[1]IRIT [2]Continental

Toulouse, France

June 27, 2017
WCET Workshop
Dubrovnik, Croatia

Cost of a Bug

Relative cost of a bug (according to IBM)

Design phase	1
After development	5
After deployment	100

'The cost to fix an error found after product release was four to five times as much as one uncovered during design, and up to 100 times more than one identified in the maintenance phase.'

— IBM Systems Sciences Institute

Cost of a Bug

Relative cost of a bug (according to IBM)

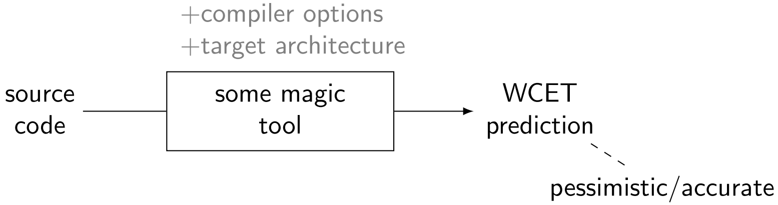
Design phase	1
After development	5
After deployment	100

'The cost to fix an error found after product release was four to five times as much as one uncovered during design, and up to 100 times more than one identified in the maintenance phase.'

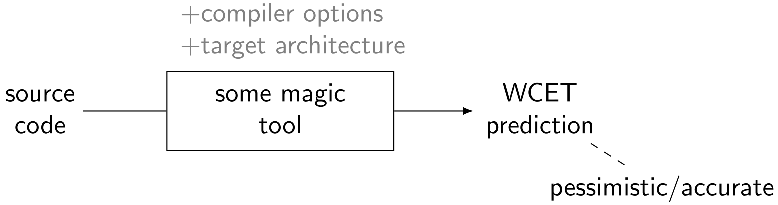
— IBM Systems Sciences Institute

→ Transfer to WCET?

What the Industry Wants



What the Industry Wants



→ Realistic expectations?

'This is a very wild idea and I doubt that it will work on real programs.'

— An anonymous reviewer

Preliminary Results Preview

Technique	Accuracy: Training set			Accuracy: Evaluation set		
Multi-Layer Perceptron	Err. 10%	24% u.	0% o.	Err. 11%	24% u.	1% o.
Random Forest	Err. 4%	3% u.	0% o.	Err. 12%	4% u.	2% o.
Linear Regression				Err. 10%	8% u.	4% o.

→ So far the results on TACLeBench are partial and disappointing

These results are due to Frédéric Fort.

Our Proposal: Use Machine Learning

Machine Learning Input: a Spreadsheet

Program	Characteristic 1	Characteristic 2	$\cdots$	**WCET**
A				1410
B				6912
⋮				

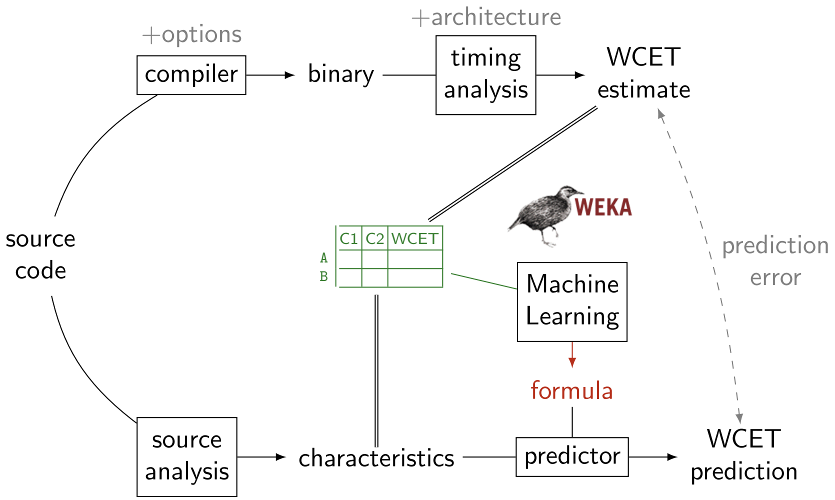
WEKA
The University
of Waikato

Machine Learning Output: a Formula

$$\textbf{WCET} \simeq f(\text{characteristics})$$

Outline of the Presentation

Learning Framework



Issue 1/3: The Learning Set

	C1	C2	···	WCET
A				
B				
⋮				

Machine Learning requires:

- Large sets (more than 1 000 programs)
- Representative sets

Candidates program sets

- Benchmarks — repr. ok, size not ok
- Industrial bank of functions — good repr., borderline size, availability issue
- Generated programs — unlimited size, doubtful repr., compilation optim. issue

Issue 2/3: The Characteristics

Choose

	C1	C2	· · ·	WCET
A				
B				

Gather

	C1	C2	· · ·	WCET
A				
B				

Machine Learning requires:

- Numerical or discrete characteristics
- Characteristics controlling the learnt attribute
  - ↪ Syntactic characteristics are insufficient

We rely on worst-case numerical metrics (details follow shortly)

Issue 3/3: Feasibility and Validation

Do it exists a set of characteristics of the source code controlling the WCET?

$\rightarrow$ We consider prediction in $\pm 20\%$ as OK

Outline of the Presentation

Source Code Analysis

	C1	C2	...	WCET
A	230	45		
B				

Worst Case Event Count Analysis

The worst case event count analysis delivers an over-approximation of the number of events by category triggered by any symbolic execution of the source program.

- Example of categories:
  - Reading a variable
  - Performing a non-trivial multiplication
  - "Branching" back to the loop head
- A mapping from category to number is called a "metric"

Playing with Metrics

- Sequential code gives one metric
- Alternatives should be combined with care
  - $m_1 \sqcup m_2$ is a sound but coarse approximation of "m_1 or m_2"

$$\begin{bmatrix} Addition \mapsto 3 \\ Product \mapsto 4 \end{bmatrix} \sqcup \begin{bmatrix} Addition \mapsto 4 \\ Division \mapsto 2 \end{bmatrix} = \begin{bmatrix} Addition \mapsto 4 \\ Product \mapsto 4 \\ Division \mapsto 2 \end{bmatrix}$$

  - If neither $m_1 \prec m_2$ nor $m_2 \prec m_1$ we continue with $\{m_1, m_2\}$
    - The analysis is disjunctive (complexity issue)
    - Using bounds on the categories costs, we strengthen the $\prec$ relation

$$\begin{bmatrix} Jump \mapsto 11 \\ Addition \mapsto 2 \end{bmatrix} \prec \begin{bmatrix} Jump \mapsto 14 \\ Product \mapsto 5 \end{bmatrix}$$

- Loop bounds are needed
- Eventually $\bigsqcup$ is used to get a unique metric

Categories retained (so far)

Our implementation of the analysis considers:

Family	Category	Optimistic	Pessimistic
Operations	Simple	0.5	1
	Multiplication	1	5
	Division	1	10
Control	Unconditional branch	0.5	1
	Conditional branch	1	1
	Computed branch	1	4
	Call	0.5	10
Memory	Address setting	0.5	1
	Load	2	20
	Store	2	20

Outline of the Presentation

Experiments Setting

- Programs are compiled with `gcc`
  - ↪ The target is ARM
- Worst Case Event Count analysis is performed by `oRange`
- WCET analysis is performed by `OTAWA`
  - ↪ The model is a simple ARMv5

Program Sets Involved

- Training set (10 000)
  ↪ Generated randomly (with `if` and `for` statements)
- Evaluation set (5 000)
  ↪ Generated the same way
- TACLeBench (23)
  ↪ Sequential part of the benchmark

Learning Techniques Explored

- Multi-Layer Perceptron
  - ↪ State-of-the-art neural network
- Random Forest
  - ↪ Huge decision tree
- Linear Regression
  - ↪ Best linear combination of the characteristics

Accuracy Evaluation

- Applying the learnt formula on a set of programs gives statistics

Correlation coefficient	0.9961
Mean absolute error	187.9878
Root mean squared error	490.4514
Relative absolute error	4.2395
Root relative squared error	8.972
Total Number of Instances	10000
Underestimations	283
Overestimations	0

- We make the following summaries:

Err. 4%	
3% u.	0% o.

Current Accuracy of our Approach

Technique	Accuracy: Training set			Accuracy: Evaluation set		
	Err.	u.	o.	Err.	u.	o.
Multi-Layer Perceptron	Err. 10%	24% u.	0% o.	Err. 11%	24% u.	1% o.
Random Forest	Err. 4%	3% u.	0% o.	Err. 12%	4% u.	2% o.
Linear Regression				Err. 10%	8% u.	4% o.

→ So far the results on TACLeBench are partial and disappointing

These results are due to Frédéric Fort.

Linear Regression Formula

$$
\begin{aligned}
\textbf{WCET} = \ & 0.0506 * \text{SimpleOp} \\
+ \ & 2.2557 * \text{Mult} \\
+ \ & 0 * \text{Div} \\
+ \ & 1.0391 * \text{CondBr} \\
+ \ & 2.638 * \text{UncondBr} \\
+ \ & 0 * \text{CalcBr} \\
+ \ & 0 * \text{Call} \\
+ \ & 0 * \text{Return} \\
+ \ & 1.9445 * \text{Address} \\
+ \ & 0 * \text{Load} \\
+ \ & 0 * \text{Store} \\
+ \ & 18.4446
\end{aligned}
$$

Related Work

Gustafsson et al.[1,2] follow the same objective with different tools

- Measurement-based approach
  - ↪ No need for hardware model but need the hardware itself
- Ad-hoc learning techniques
- Evaluation on the Mälardalen benchmark gives:
  - Accurate approach:

Err. 8%	
4% u.	9% o.

  - Pessimistic approach:

Err. 31%	
0% u.	52% o.

[1] Jan Gustafsson et al. "Approximate Worst-Case Execution Time Analysis for Early Stage Embedded Systems Development". In: **SEUS**. 2009.

[2] Peter Altenbernd et al. "Early execution time-estimation through automatically generated timing models". In: **Real-Time Systems** (2016).

Conclusion

Work in Progress

- We seek early WCET predictions through Machine Learning
- We proposed a source analysis for retrieving program metrics
  - ↪ Required for both learning and predicting
- Several question are opened
  - Relevant characteristics
  - Best learning technique