AN EFFORT ALLOCATION METHOD TO OPTIMAL CODE SANITIZATION FOR QUALITY-AWARE ENERGY EFFICIENCY IMPROVEMENT

MARCO BESSI (m.bessi@castsoftware.com)
GABRIELLA CARROZZA (gabriella.carrozza@accenture.com)
ROBERTO PIETRANTUONO (roberto.pietrantuono@unina.it)
STEFANO RUSSO (stefano.russo@unina.it)

3rd International Workshop on Measurement and Metrics for Green and Sustainable Software Systems (MeGSuS '16)

Ciudad Real, 07/09/2016

AGENDA

- **Problem Analysis**
- ASA vs Energy consumtion relationship
- ASA vs Defects relationship
- Optimal code sanitization
- Conclusion

PROBLEM ANALYSIS

Software energy efficiency

Infrastructural levels

Software energy efficiency is related to the computational cost to execute but also to the correctness of itself (a fail in the execution need a retry the consume additional energy).

- code energy efficiency and defectiveness are related to software quality, checked with an Automatic Static Analysis (ASA) tool.

Method proposed:

Prediction: Use relations between ASA rules and energy/defectiveness attribute to rank applications accordingly to criticality.

Optimization: Allocate effort for code improvement.

Multi-objectives optimization model across applications to sanitize code in order to reduce the cost to fix violations, to maximize the impact on energy and to reduce the defectiveness

AGENDA

- Problem Analysis
- **ASA vs Energy consumtion relationship**
- ASA vs Defects relationship
- Optimal code sanitization
- Conclusion

ASA VS ENERGY EFFICIENCY RELATIONSHIP

$$E_{TOT} \cong E_{idle} + E_{CPU} + E_{DB} + E_{NET} \approx_{CPU} E_{idle} + E_{CPU} \cong P_{idle} * t_{ex} + \gamma * \beta$$

$$\cong P_{idle} * t_{ex} + [\gamma(u,t,e,d)] * \beta$$

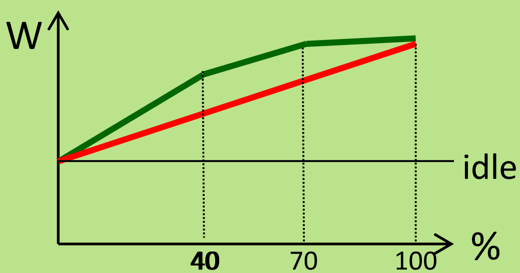
For computational-intensive ERP transactions, the energy consumption is given by the product of γ and β metrics

Dimension	Influence on γ
Number of concurrent users u	YES
Transaction typology t	YES
ERP application e	YES
DB population d	YES
OS o	No
Hardware h	No

CPU

Energy cost of computational resources

$$\boldsymbol{\beta} = \Delta\,\%_{CPU} * f$$

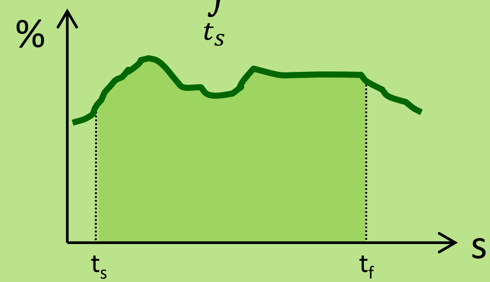


HW dependent

- Obtained by means of benchmark tests or datasheets

Resource consumption

$$\boldsymbol{\gamma}(\mathbf{u},\mathbf{t},\mathbf{e},\mathbf{d}) = \left(\int_{t_s}^{t_f} \%_{CPU}\, dt * \boldsymbol{\omega}(\boldsymbol{u},\boldsymbol{t},\boldsymbol{e},\boldsymbol{d})\right)/f$$



Application dependent

- Obtained with application stress tests

Normalized with **scalability metric $\boldsymbol{\omega}$**

[1] M. Bessi, E. Capra, C. Francalanci. "A benchmarking methodology to assess the energy performance of MIS applications". ICIS, 2013

ASA VS ENERGY EFFICIENCY RELATIONSHIP

Problems related to the γ metric:

Goodness of test cases	The test cases have to represent the real industrial usage
No hint for optimization	Developers have no direct hint for source code optimization
Too late results	Results are provided during the testing at the end of development lifecycle

Thus, we explore performance indicators from ASA (CAST) looking at spearman correlations between ASA results and γ metric.

Programming rules	**Spearman correlation #violation & γ metric**
Avoid using Driver Manager	**0.93**
Avoid the use of Instanceof inside loops	**0.87**
Avoid using Hash Table	**0.85**
Avoid String initialization with String object	**0.85**
Avoid String concatenation in loops	**0.75**
Avoid using Dynamic instantiation	**0.72**

AGENDA

- Problem Analysis
- ASA vs Energy consumtion relationship
- **ASA vs Defects relationship**
- Optimal code sanitization
- Conclusion

ASA VS DEFECTIVENESS RELATIONSHIP

User perceives system's failures: one could have a very poor (internal) code quality in terms of violations, but a good user experience.

- ASA violations are detected during coding
- Less expensive to resolve

- Defects are detected during testing or operations
- More expensive to resolve

Programming rules	**Spearman correlation #violations & defects**
Metrics rules	**0.85**
Naming convention rules	**0.46**
Possible bugs	**0.45**
Coding convention rules	**0.45**
Formatting rules	**0.45**
Memory and resource	**0.44**

[2] G. Carrozza, M. Cinque, U. Giordano, R. Pietrantuono, and S. Russo, "Prioritizing correction of static analysis infringements for cost-effective code sanitization, SER&IP'15, IEEE Press, 2015.

AGENDA

- Problem Analysis
- ASA vs Energy consumtion relationship
- ASA vs Defects relationship
- **Optimal code sanitization**
- Conclusion

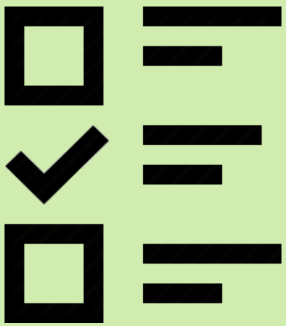
OPTIMAL CODE SANITIZATION

Prediction

Machine learning algorithms are run to build predictive models.

- Spot the most relevant applications
- Spot the most relevant ASA rules

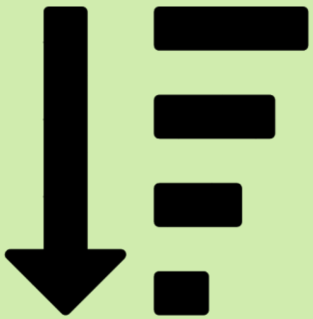
from defectiveness and energy point-of-view



Optimization

Output of prediction phase is exploited to orient the fixing fo ASA rule violations in term of:

- Which applications should be prioritized
- Which ASA rules should be prioritized



OPTIMAL CODE SANITIZATION: PREDICTION

1. **Retrieving data from ASA reports** about a set of applications, with defects and energy consumption measurement available

2. **Training and test of the prediction models**

3. **Prediction** on application without defects and energy informations (but with ASA violations).
   - Output: list of applications rated with expected energy and defectiveness.
   - Ranking based algorithm associate a ranking score p_j(for defects) and q_j(for energy)
   - Normalize the ranking in [0,1]: w_{d_j} (for defects) and w_{e_j} (for energy)

$$w_{d_j} = \frac{p_j - \min(p_j)}{\max(p_j) - \min(p_j)} \qquad w_{e_j} = \frac{q_j - \min(q_j)}{\max(q_j) - \min(q_j)}$$

4. **Rule rating**. For each rule i is assigned a score s_i (used for defects) and h_i (used for energy)
   - Score are normalized in [0,1]: ρ_{d_i} (for defects) and ρ_{e_i} (for energy)

$$\rho_{d_i} = \frac{s_i - \min(s_i)}{\max(s_i) - \min(s_i)} \qquad \rho_{e_i} = \frac{h_i - \min(h_i)}{\max(h_i) - \min(h_i)}$$

Prediction information (w_{d_j}, ρ_{d_i} and w_{e_j}, $\rho_{\boldsymbol{e_i}}$) is used to parametrize the optimization model.

OPTIMAL CODE SANITIZATION: OPTIMIZATION

Objective:

Using a multi-objective optimization approach, we suggest the number and type of violations to remove and the applications from which they should be removed in order to get the best tradeoffs among:

- maximizing the expected quality,
- maximizing the expected energy efficiency ,
- minimizing the expected cost.

$$Max!\ Q = \sum_{j}^{n}\left[\sum_{i=1}^{m}(x_{i,j} * \rho_{d_i} * S_{d_i} * w_{d_j})\right]$$

$$Max!\ E = \sum_{j}^{n}\left[\sum_{i=1}^{m}(x_{i,j} * \rho_{e_i} * S_{e_i} * w_{e_j})\right]$$

$$Min!\ \ C = \sum_{j}^{n}\left[\sum_{i=1}^{m}(x_{i,j} * C_{fix_i})\right]$$

where:

- S_{d_i} and S_{e_i}represent the severity assigned to each rule to give priorities to the removal of violations
- $x_{i,j}$ represent the nuober of violations of the i^{th} type that algorithm propose to eliminate from application j

Output: Matrix reporting the #violations that should be removed for each type and for each application, the estimated cost of removal, the expected quality in terms of defects and the expected energy consumption.

AGENDA

- Problem Analysis
- ASA vs Energy consumtion relationship
- ASA vs Defects relationship
- Optimal code sanitization
- **Conclusion**

CONCLUSION: ROADMAP

1

Validate the prediction and the optimization steps separately.

- Prediction – on extended set of test applications to improve generality and extend set of metrics;
- Optimization – refine the module and test it on real case scenario.

2

The whole methodology is experimented on real case studies.

QUESTIONS

Marco Bessi

+39 342 919 9530
m.bessi@castsoftware.com

Solution Delivery Consultant
CAST Italia
Via San Vittore 49
20123 Milano (Italy)

Instead of controlling the environment
for the benefit of the population,
perhaps it's time we control the population
to allow the survival of the environment.
[David Attenborough - English broadcaster and naturalist]