GREEN INDEXES USED IN CAST TO MEASURE THE ENERGY CONSUMPTION IN CODE

Marco Bessi
Solution delivery consultant @ CAST
Marco Bessi

- Solution delivery consultant
- CAST green evangelist

- Computer science PhD on software energy efficiency estimation and optimization through Java bytecode reengineering

- Collaborations with:
GREEN ICT FOR A SUSTAINABLE WORLD
Introduction

- Gartner [1] said that 2% of total CO$_2$ emissions is related to ICT

- Energy in Datacenter [2]:

  - 56% server
  - 14% others (network, light, ...)
  - 30% cooling

Introduction

- In the last 10 years, IT systems become bigger and bigger, with a consequent growth of the IT energy consumptions

1. Energy costs have dramatically increased

2. Energy requirements represent one of the scalability issues of datacenters

3. IT contributes strongly to the CO$_2$ emissions
Energy efficiency of Datacenters

- Servers are rarely idle or working at full capacity [3]
  - Most of the time are between 10-50%
  - Idle or low capacity is a waste of energy

Green computing

- San Murugesan [4] define the «Green computing» as:

  «the study and practice of designing, manufacturing, using, and disposing of computers, servers, and associated subsystems efficiently and effectively with minimal or no impact on the environment»

Green computing

Select the right «path»: the automotive analogy.

- Fixed the car, and then its unit consumption (litres/km)
- Fixed two point A and B
- Select the right path!

- Fixed the server that runs your application
- Fixed the functional behavior
- Wrote the most efficient algorithm
Choosing a «good» software enables energy savings

PUE (Power Usage Effectiveness) = \( \frac{\text{TotalPower}}{\text{ITPower}} \)

Infrastructure amplifies server efficiency and inefficiency by a factor equal to the PUE (Italy best-in-class = 1.2; Italy mean = 1.8)

APR (Application Power Ratio) = \( \frac{\text{ServerPower}}{\text{ServerIdle}} \)

Applications that satisfy the same functional requirements may consume significantly different amounts of energy (on average 30%) (CPU intensive = 1.7; Transactional application = 1.4)
Know your enemy

86% ICT departments

80% companies

NOT CO₂

NOT

It is hardly impossible to optimize what it is NOT monitored

CAST
CAST company overview

**SAM**

- CAST is an independent software vendor that is a pioneer and world leader in Software Analysis and Measurement (SAM)

**1990**

- Founded in 1990, CAST is listed on Euronext (CAS.PA) and serves IT-intensive Global 2000 enterprises and public sector institutions worldwide. CAST has a presence in the Americas, EMEA, and India.

**FACT BASED**

- CAST introduces fact-based transparency into software asset management, application development, maintenance and sourcing to transform it into a management discipline.

Our mission is to provide a standard unit of measure for those who build, buy or sell software

Measure what can be measured, and make measurable what cannot be measured.

Galileo Galilei (1564-1642)
CAST and its ecosystem

250+ ENTERPRISE CUSTOMERS COUNT ON CAST

GLOBAL SYSTEM INTEGRATORS RELY ON CAST

GLOBAL PRESENCE
USA, Germany, UK, France, Belgium, Italy, Spain, India

THE UNIT OF MEASURE FOR THOSE WHO BUILD, BUY OR SELL SOFTWARE

MARKET LEADER, PURE PLAYER, GLOBAL
NYSE Euronext

at&t  SANOFI  vodafone  Allianz
WELLS FARGO  RWE  Deutsche Telekom
CREDIT SUISSE  UnitedHealthcare
DB  REPSOL  AMERICAN EXPRESS
HSBC  BNP PARIBAS
UniCredit  Broadridge
Enterprise-grade analysis requires a 3-tier approach

**Architecture Compliance**

- Intra-technology architecture
- Intra-layer dependencies
- Module complexity & cohesion
- Design & structure
- Inter-program invocation
- Security Vulnerabilities

**Program Level**

- Code style & layout
- Expression complexity
- Code documentation
- Class or program design
- Basic coding standards

**Module Level**

- Intra-technology architecture
- Intra-layer dependencies
- Module complexity & cohesion
- Design & structure
- Inter-program invocation
- Security Vulnerabilities

**System Level**

- Integration quality
- Architectural compliance
- Risk propagation simulation
- Application security
- Resiliency checks
- Transaction integrity
- Function point & EFP measurement
- Effort estimation
- Data access control
- SDK versioning
- Calibration across technologies

---

3rd International Workshop on Measurement and Metrics for Green ans Sustainable Software Systems (MeGSuS’16)
CAST is the only SAM solution that inspects and verifies software in accordance to CISQ’s recommendations.

### CISQ Quality Characteristic Measures

<table>
<thead>
<tr>
<th>Security</th>
<th>Ability to prevent unauthorized intrusions and data theft</th>
<th>Damages, customer confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Ability to avoid outages and to recover operations quickly</td>
<td>Damages, lost revenue, customer loss</td>
</tr>
<tr>
<td>Performance Efficiency</td>
<td>Ability to avoid response degradation, resource overuse</td>
<td>Lost customers, operating cost</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Ability to understand and modify software quickly</td>
<td>Cost of ownership, time to market</td>
</tr>
</tbody>
</table>

http://it-cisq.org/standards/automated-quality-characteristic-measures
CISQ focuses on the software flaws that matter

- Software Risk Prevention:
  - Focus on critical violations that matter
  - Focus resources on areas of highest impact not highest number of flags

"Tracking programming practices at the Unit Level alone may not translate into the anticipated business impact,...most devastating defects can only be detected at the System Level."
- OMG

<table>
<thead>
<tr>
<th>Business Characteristics</th>
<th>Good Coding Practices</th>
<th>Good Architectural Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unit-Level</td>
<td>Technology/System Levels</td>
</tr>
<tr>
<td>RELIABILITY</td>
<td>Protecting state in multi-threaded environments</td>
<td>Multi-layer design compliance</td>
</tr>
<tr>
<td></td>
<td>Safe use of inheritance and polymorphism</td>
<td>Software manages data integrity and consistency</td>
</tr>
<tr>
<td></td>
<td>Resource bounds management, Complex code</td>
<td>Exception handling through transactions</td>
</tr>
<tr>
<td></td>
<td>Managing allocated resources, Timeouts</td>
<td>Class architecture compliance</td>
</tr>
<tr>
<td>PERFORMANCE EFFICIENCY</td>
<td>Compliance with Object-Oriented best practices</td>
<td>Appropriate interactions with expensive or remote resources</td>
</tr>
<tr>
<td></td>
<td>Compliance with SQL best practices</td>
<td>Data access performance and data management</td>
</tr>
<tr>
<td></td>
<td>Expensive computations in loops</td>
<td>Memory, network and disk space management</td>
</tr>
<tr>
<td></td>
<td>Static connections versus connection pools</td>
<td>Centralized handling of client requests</td>
</tr>
<tr>
<td></td>
<td>Compliance with garbage collection best practices</td>
<td>Use of middle tier components vs. procedures/DB functions</td>
</tr>
<tr>
<td>SECURITY</td>
<td>Use of hard-coded credentials</td>
<td>Input validation</td>
</tr>
<tr>
<td></td>
<td>Buffer overflows</td>
<td>SQL injection</td>
</tr>
<tr>
<td></td>
<td>Missing initialization</td>
<td>Cross-site scripting</td>
</tr>
<tr>
<td></td>
<td>Improper validation of array index</td>
<td>Failure to use vetted libraries or frameworks</td>
</tr>
<tr>
<td></td>
<td>Improper locking</td>
<td>Secure architecture design compliance</td>
</tr>
<tr>
<td></td>
<td>Uncontrolled format string</td>
<td></td>
</tr>
<tr>
<td>MAINTAINABILITY</td>
<td>Unstructured and duplicated code</td>
<td>Duplicated business logic</td>
</tr>
<tr>
<td></td>
<td>High cyclomatic complexity</td>
<td>Compliance with initial architecture design</td>
</tr>
<tr>
<td></td>
<td>Controlled level of dynamic coding</td>
<td>Strict hierarchy of calling between architectural layers</td>
</tr>
<tr>
<td></td>
<td>Over-parameterization of methods</td>
<td>Excessive horizontal layers</td>
</tr>
<tr>
<td></td>
<td>Hard coding of literals</td>
<td>Excessive multi-tier fan-in/fan-out</td>
</tr>
<tr>
<td>NUMBER OF ISSUES</td>
<td>90% of violations</td>
<td>10% of violations</td>
</tr>
<tr>
<td>BUSINESS IMPACT</td>
<td>52% of repair workload</td>
<td>48% of repair workload</td>
</tr>
<tr>
<td></td>
<td>10% of production downtime</td>
<td>90% of production downtime</td>
</tr>
</tbody>
</table>

"Tracking programming practices at the Unit Level alone may not translate into the anticipated business impact,...most devastating defects can only be detected at the System Level."
- OMG

- 90% Downtime caused by system-level flaws!
- 92% Of all defects
- 52% Of total repair effort
- 10% UNIT LEVEL FLAWS
CAST Application Intelligence Platform (AIP)

Deep Insight into Critical Applications:

1. **System-Level* Analysis**
   Source code, transactions, data structure

   "Architectural software flaws lead to 90% of the production issues."
   Dr. R. Soley, OMG/ Ph.D MIT

2. **Best practices & industry Standards Checks**
   Architectural integrity and critical violations

3. **Accurate Analytics & Benchmarks**
   - APP ANALYTICS DASHBOARD
     Business relevant Analytics
   - APP ENGINEERING DASHBOARD
     Critical Structural Flaws, Action Plans and eLearning
Risk analytics, a management scorecard

Portfolio-level risk and saving opportunities

Likelihood of outage, data integrity or reliability issues

Resource consumption, scalability and performance issues

Security issues and high likelihood of breaches

Adaptability to changing regulations and business needs

Ramp up difficulties for newcomers

Standardized units of ADM work with consistent technical and functional sizing for productivity measurement

Cost to restore applications back to healthy state

Hard-to-find structural flaws that may lead to software catastrophes
Metric calculation

- For each rule a grade is computed.
- The grade is between $[1,4]$, where 1 represent the very high risk and 4 the low risk.
  - Below 3, risks and costs increase.
- Each rule has different thresholds accordingly to its risk importance.

\[
\text{Compliance Ratio} = \frac{\text{successful checks}}{(\% \text{ of Successful checks})} \times 100
\]

\[
\text{successful checks + failed checks}
\]

![Compliance Ratio Graph](image)

- Very high risk
- High risk
- Moderate risk
- Low risk

Threshold #1: E.g., 10%
Threshold #2: E.g., 70%
Threshold #3: E.g., 90%
Threshold #4: E.g., 99%
Quality Model, a key for unbiased measurement

Quality Metrics Subset
- SQL Complexity Distribution
- Class complexity (Inh. depth)
- Class complexity (Inh. width)
- Artifacts having recursive calls
- Method complexity (control flow)
- Multiple artifacts inserting data on the same SQL table
- Coupling Distribution
- File conformity
- Dead code
- Structuredness
- Controlled data access
- Empty code
- Modularity
- Encapsulation conformity
- Inheritance
- Package naming
- Class naming
- Interface naming
- Package comment
- Class comment
- Method comment
- Package size
- Class size (methods)
- Interface size

Criteria
- Complexity
- Architecture
- Programming Practices
- Naming Conventions
- Documentation

Health Factors
- Efficiency
- Robustness
- Security
- Transferability
- Changeability
- Maintainability

Application Quality
- Green IT Index

Risk drivers
- Cost drivers

1500+ architectural and language-specific code checks
CAST GREEN IT INDEX
The problem is simple, the solution is less so

- CAST has developed a four-pronged approach to guiding action based on simple components
  - Efficiency in operations – do the most with the least effort
  - Complexity – Simple tends to take less time, less power, less support
  - Programming – make the right choices
  - Security – Prevent intrusion

Energy can be saved by making software more efficient
The problem is simple, the solution is less so

- Lack of **EFFICIENCY** in IT operations waste energy
  - Unnecessary CPU cycles means unnecessary kWh consumption
  - IT operations run on electricity
  - kWh production leads to CO₂ emission (or equivalent)
The problem is simple, the solution is less so

- **COMPLEXITY** adds time, consumes power and delivers dubious improvements
The problem is simple, the solution is less so

- **PROGRAMMING** makes it so
  - People have been used to ever-growing computing resources, omitting the impact on the environment through the energy consumption
    - Resulting in software that is far from optimized
The problem is simple, the solution is less so

- SECURITY should be a pervasive thought in all
How CAST support energy efficient software

- CAST offers a solution based on his Assessment Model
  - The Green IT Index is a Business Criterion that aggregates Quality Rules from existing high-impact Efficiency-related Technical Criteria
  - as well as from most-severe Robustness-related Technical Criteria to account for wasted resources when application functioning has been compromised and require a restart/recovery

- CAST released this first version of this Green IT Index to leverage the expertise and experience of select customers in order to fine-tune its exact composition. Our experience has shown that it is more relevant than ever.

- CAST can show how to make software more efficient and save both “Green” and ($) “Green”
Energy can be saved with software more efficient

- CAST support energy efficient software to identify pieces of software that could be optimized to require less CPU resources.
  But:
  - This is NOT about software pieces that require a lot of CPU resources
  - This is about software pieces that WASTE a lot CPU resources

- Many other factors intervene in Software energy consumption
  - Current indicator only covers the aspects that can be solved by evolving the Software code
  - It excludes Software design, deployment, …

- This indicator does not try to provide a CO2eq
  - CO2eq ⇔ kWh is still a disputed and volatile ratio
  - E.g.: 0.09 kg CO2eq ⇔ French kWh / 0.46 kg CO2eq ⇔ European kWh / …
  - CPU cycle ⇔ kWh is a volatile ratio
  - E.g.: running Unix environment on a z/Architecture / on IA-32 / on IA-64…
AAD Demo: Green IT Index

Green IT Index detail

- Select the application that you want to investigate

Application Green IT Index

The Green IT Index of your applications is displayed in the bar graph above. Green IT index range from 1.00 to 4.00 with 4.00 being the best possible score. Green IT Index helps to identify pieces of software that could be optimized to require less CPU resources. Current index only covers the aspects that can be solved by evolving the software code.

You can drag sections to zoom into the graph.
Green IT Index detail

- Investigate which rules impact the Green IT Index

---

<table>
<thead>
<tr>
<th>Rules</th>
<th>Weight</th>
<th>% Compliance</th>
<th>Violations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avoid call to AcceptChanges in a loop</td>
<td>1</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>Avoid doing select on Datarable in loop</td>
<td>1</td>
<td>82%</td>
<td>5</td>
</tr>
<tr>
<td>Avoid empty catch blocks</td>
<td>1</td>
<td>99%</td>
<td>70</td>
</tr>
<tr>
<td>Never perform C-style cast between incompatible Class pointers</td>
<td>1</td>
<td>97%</td>
<td>68</td>
</tr>
<tr>
<td>Avoid using ‘delete this’</td>
<td>1</td>
<td>99%</td>
<td>1</td>
</tr>
<tr>
<td>Prefer UNION ALL to UNION</td>
<td>1</td>
<td>0%</td>
<td>1</td>
</tr>
<tr>
<td>Avoid catching an exception of type Exception, RuntimeException, or Throwable</td>
<td>1</td>
<td>86%</td>
<td>1,543</td>
</tr>
<tr>
<td>Avoid instantiations inside loops</td>
<td>1</td>
<td>97%</td>
<td>270</td>
</tr>
<tr>
<td>Avoid missing default in switch statements</td>
<td>1</td>
<td>98%</td>
<td>155</td>
</tr>
<tr>
<td>Avoid method invocation in a loop termination expression</td>
<td>1</td>
<td>99%</td>
<td>83</td>
</tr>
</tbody>
</table>
Green IT index formula

- The Green IT Index is based on
  - 5 Critical Technical from Efficiency
  - 3 Critical Technical from Robustness

- Efficiency
  - Efficiency – Expensive Calls in Loops
  - Efficiency – Memory, Network and Disk Space Management
  - Efficiency – SQL and Data Handling Performance
  - Complexity – Dynamic Instantiation
  - Complexity – SQL Queries

- Robustness
  - Programming Practices – Error and Exception Handling
  - Programming Practices – Unexpected Behavior
  - Secure Coding – Time and State
Questions

- CAST wants to continue to investigate into this approach.
- You want to share your experience?

Marco Bessi
m.bessi@castsoftware.com

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