DIPARTIMENTO DI INFORMATICA SISTEMISTICA E COMUNICAZIONE

Università degli studi di Milano-Bicocca
Corso di Laurea Magistrale in Informatica
Anno accademico 2016/2017

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Green smell identification for software energy efficiency optimization

Relatore: Prof.ssa Francesca Arcelli Fontana
Co-relatore: Dott. Marco Bessi

Corrado Ballabio - 764452

28 marzo 2018

Green IT

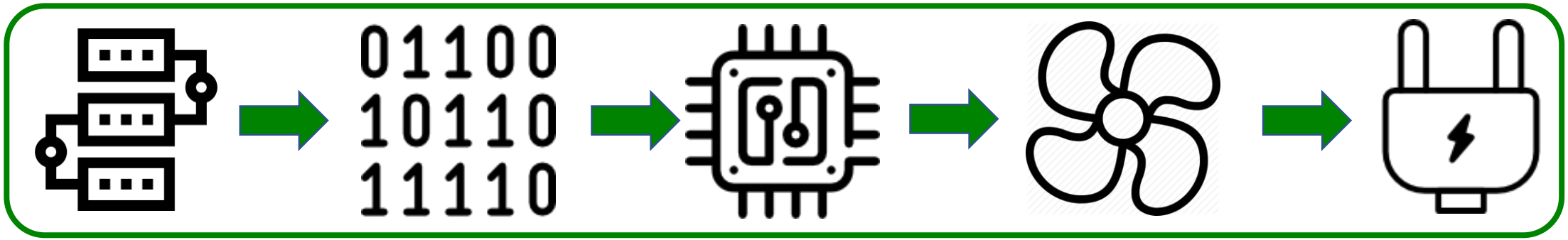
"is the study and practice of designing, manufacturing, using, and disposing of computers, servers, and associated subsystems - such as monitors, printers, storage devices, and networking and communications systems - efficiently and effectively with minimal or no impact on the environment." [1]

IT as an enabler of green governance

Eco-compatible management of IT products lifecycle

Energy efficiency of IT

Also software takes part in energy consumption:



[1] San Murugesan. Harnessing green it: Principles and practices. IT professional, 10(1), 2008.

Thesis outline

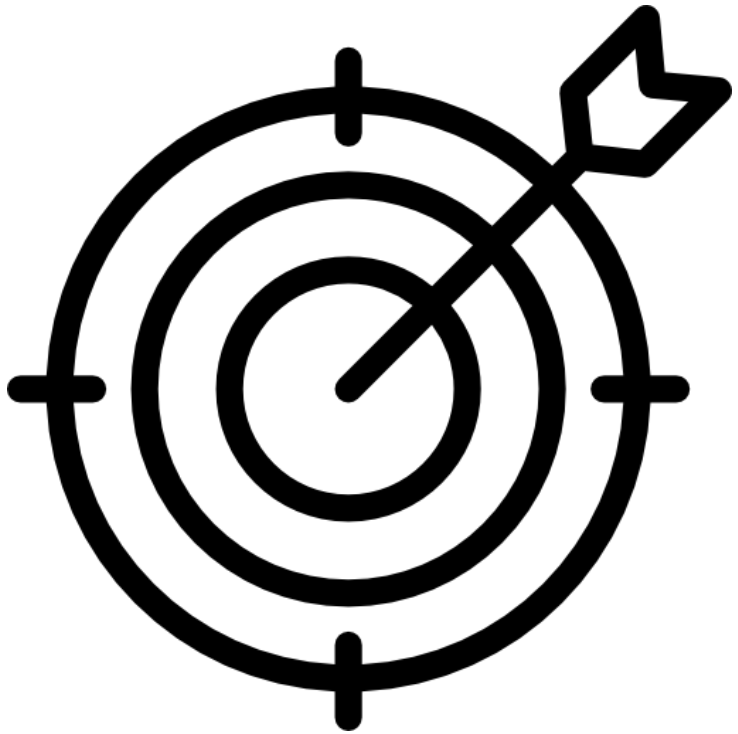
Main works assess the energetic impact that software has on embedded systems [2] or mobile devices [3]

No methodology that quantifies software energy efficiency

RQ1 **Wich of the suspected patterns are *green smells*?**

RQ2 **What are the variables that influences the energy consumption?**

Definition of a *greenability* metric for assessing energy efficiency of sample applications



[2] Antonio Vetro, Luca Ardito, Giuseppe Procaccianti, and Maurizio Morisio. Definition, implementation and validation of energy code smells: an exploratory study on an embedded system. 2013.
[3] Marion Gottschalk, Jan Jelschen, and Andreas Winter. Saving energy on mobile devices by refactoring. In EnviroInfo, pages 437–444, 2014..

Research of the patterns

- Efficiency, performance, robustness, bad practices and dodgy code rules from different sources:
  - CAST Quality Rules documentation[1]
  - FindBugs [2] and PMD[3] ruleset
- Some rule selection criteria…

53 identified patterns

- 6 Expensive Calls in Loops
- 11 Expensive Object Instantiation
- 2 Garbage Collector calls
- 19 Poor Programming Choices
- 15 DB calls

- Every pattern is implemented as a pair of functions
- Repeated 1 million times, 50 samples collected

[1] http://doc.castsoftware.com/display/DOC82/Metrics+and+Quality+Rules+-+details
[2] http://findbugs.sourceforge.net
[3] https://pmd.github.io/pmd-6.0.0/index.html

RQ1: Energy results

Pattern name	Impact (in %)	rank
callPropertiesThatCloneValuesInLoop	-116.73	9
indirectExceptionHandlingInsideLoops	-130.04	9
indirectStringConcatenationInLoops	-56.77	9
instantiationInsideLoop	-22.6	4
stringConcatenationInLoop	-32.98	7
existsIndipendentClause	-77.3	9
sqlQueriesInsideLoop	-124.30	9
boxedPrimitiveToString	-34.73	7
boxingImmediatelyUnboxed	-20.86	4
declareStaticMethod	-12.35	4
dynamicInstantiation	-107.52	9
instantiatingBoolean	-18.72	4
needlessInstantiation	-5.44	2
newObjectForGetClass	-9.7	2
numberCtor	-12.73	4
randomUsedOnlyOnce	-119.69	9
stringInitializationWithObject	-24.88	4
nonShortCircuit	-26.96	4
useArraysAsList	-155.73	9
usingHashtable	-56.08	9
usingRemoveAllToClearCollection	-36.45	7

RQ1: which of the patterns under test represent an energy hotspot (i.e. which are green smells)?

- Measurements of energy consumption are performed with jRAPL [4]
- Green smells are identified according to the Mann-Whitney test results and the impact that the resolution has on the consumption

21 patterns out of 53 are proved to be *green smells*

- *Expensive calls in loops* and *expensive object instantiation* are the less energy efficient categories

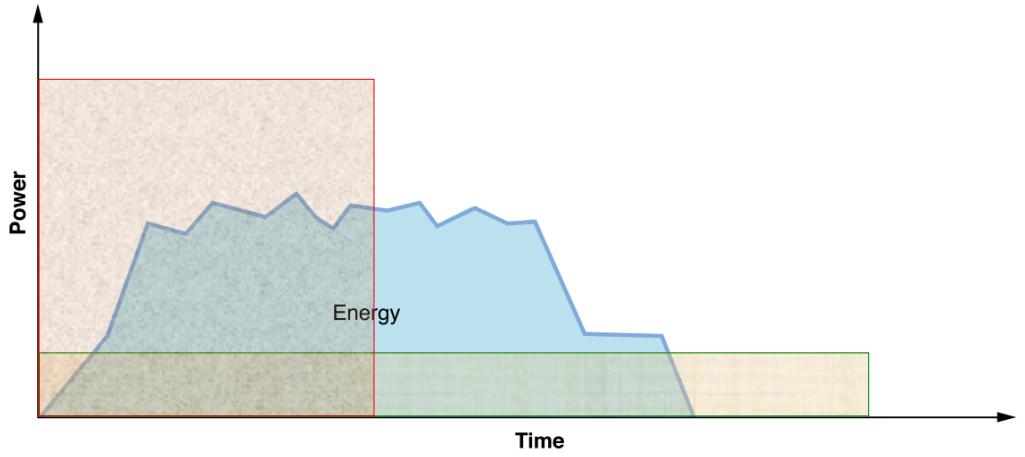
[4] Kenan Liu, Gustavo Pinto, and Yu David Liu. Data-oriented characterization of application-level energy optimization. In FASE, pages 316–331, 2015.

RQ2: Time and resources correlation

Pattern name	Performance smell	Memory smell
callPropertiesThatCloneValuesInLoop	✓	
indirectExceptionHandlingInsideLoops	✓	✓
indirectStringConcatenationInLoops	✓	✓
instantiationInsideLoop	✓	✓
stringConcatenationInLoop	✓	✓
existsIndipendentClause	✓	
sqlQueriesInsideLoop	✓	✓
boxedPrimitiveToString	✓	✓
boxingImmediatelyUnboxed	✓	✓
declareStaticMethod	✓	✓
dynamicInstantiation	✓	
instantiatingBoolean	✓	✓
needlessInstantiation		
newObjectForGetClass	✓	✓
numberCtor	✓	✓
randomUsedOnlyOnce	✓	
stringInitializationWithObject	✓	✓
nonShortCircuit	✓	
useArraysAsList	✓	✓
usingHashtable	✓	
usingRemoveAllToClearCollection	✓	✓

RQ2: which are the main variables between execution time, resource usage and bytecode instruction, that are responsible of the increase in energy consume?

- Additional time and resource analysis identify *performance* and *resources smells*
- *uselessSubstring* is a performance smell but not a green smell
- *largeNumberOfStringConcat, formatStringNewLine* and *staticFieldCollection* are memory smells but not green smells



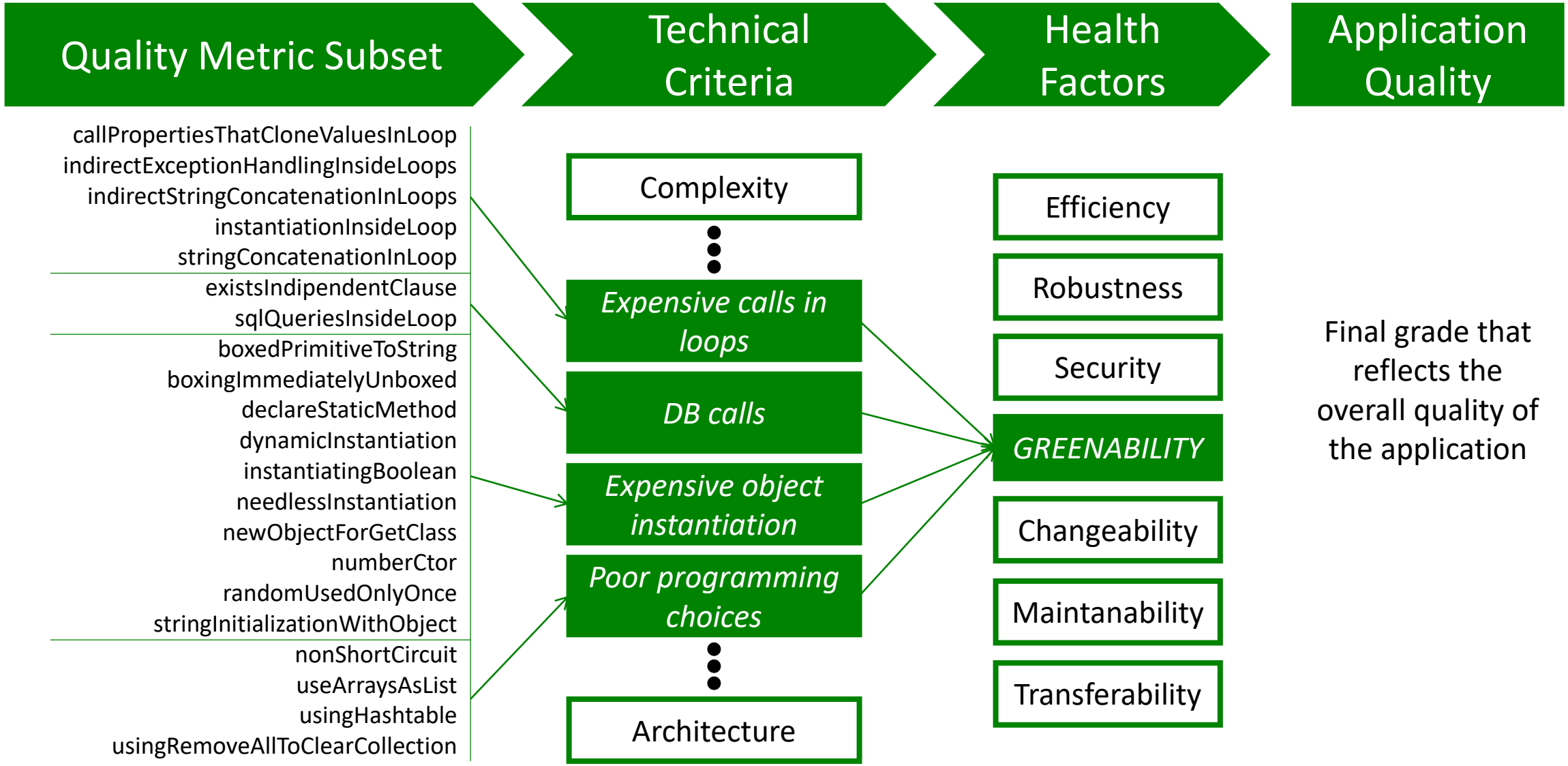
Bytecode analysis

```
public static void numberCtor_With_FindBugs();
  Code:
     0: new           #7
     3: dup
     4: bipush        11
     6: invokespecial #8
     9: astore_0
    10: return

public static void numberCtor_Without_FindBugs();
  Code:
     0: bipush        11
     2: invokestatic  #9
     5: astore_0
     6: return

public static void instantiatingBoolean_With_CAST();
  Code:
     0: new           #32
     3: dup
     4: iconst_1
     5: invokespecial #33
     8: astore_0
     9: aload_0
    10: invokevirtual #34
    13: ifeq          16
    16: return

public static void instantiatingBoolean_Without_CAST();
  Code:
     0: getstatic     #35
     3: invokevirtual #34
     6: ifeq          9
     9: return
```

- Bytecode analysis highlights a bigger presence of new and `invoke` operations
- Their presence is related to object instantiation:
  - new for allocating memory for the new instance
  - `invokespecial` for calling the Class constructor

  And to methods invocation:
  - `invokeinterface` is used to invoke a method declared within a Java interface
  - `invokestatic` calls static methods
  - `invokevirtual` for all the other cases

Greenability metric



Greenability results

- A sample analysis is performed on WebGoat[4], a deliberately flawed application used to test vulnerabilities commonly found in Java-based applications
- The metric allows an assessment of the energy efficiency of sample Java applications, and gives the ability to analyse and refactor the code from a energy efficient point of view

Business Criteria

Grade	Var.	Business Criterion
2.4	0.00%	Programming Practice
2.45	0.00%	Architectural Design
2.47	0.00%	Total Quality Index
2.48	0.00%	Changeability
2.51	0.00%	Documentation
2.54	0.00%	Robustness
2.81	0.00%	Transferability
3.28	0.00%	SEI Maintainability
3.58	0.00%	Greenability

Technical Criteria

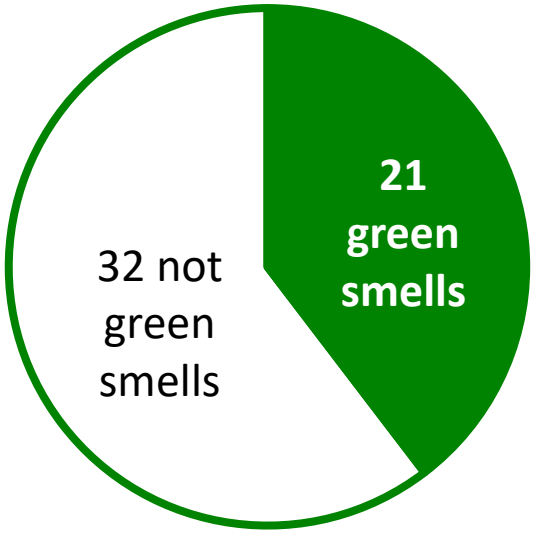
Grade	Var.	Technical Criterion	Contr.	Critical
2.68	0.00%	Greenability Efficiency - Expensive Calls in Loops	7x50%	No
3.72	0.00%	Greenability Efficiency - Expensive Object Instantia	5x50%	No
4	0.00%	Greenability Efficiency - DB calls	8x50%	No
4	0.00%	Greenability Efficiency - Poor Programming Choiche	5x50%	No

Quality Rules, Distributions and Measures

Grade	Var.	Rule Name	Contr.	Critical
1.93	0.00%	Declare as Static all methods not using instance me	4x50%	No
3.77	0.00%	Avoid using Dynamic instantiation	9x50%	Yes
3.96	0.00%	Avoid String initialization with String object (create	4x50%	No
4	0.00%	Avoid allocating a boxed primitive value just to call	7x50%	No
4	0.00%	Avoid boxing a primitive value and then immediatel	4x50%	No
4	0.00%	Avoid ineffiient Number constructor calls	4x50%	No
4	0.00%	Avoid instantiating Boolean	4x50%	No
4	0.00%	Avoid allocating an object just to call getClass() me	2x50%	No

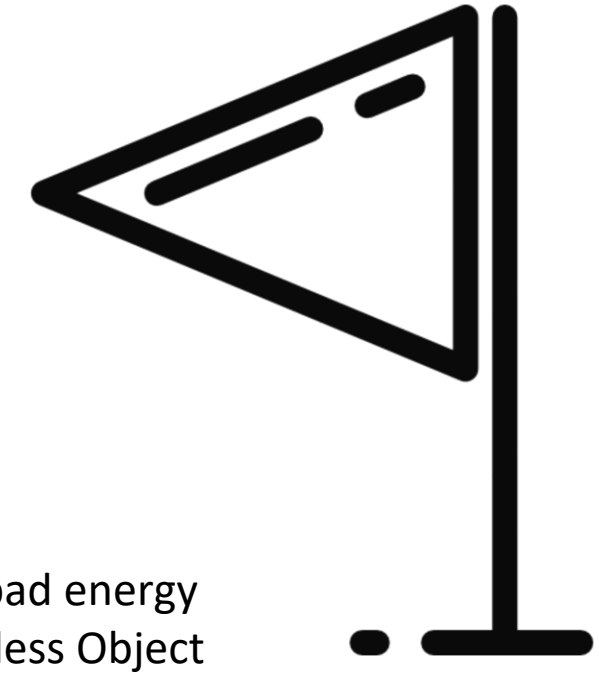
[4] https://github.com/WebGoat/WebGoat/wiki

Conclusions



RQ1: which of the patterns under test represent an energy hotspot (i.e. which are green smells)?

Green smells are mostly related to non-optimal Object instantiations and to expensive operation executed inside loops

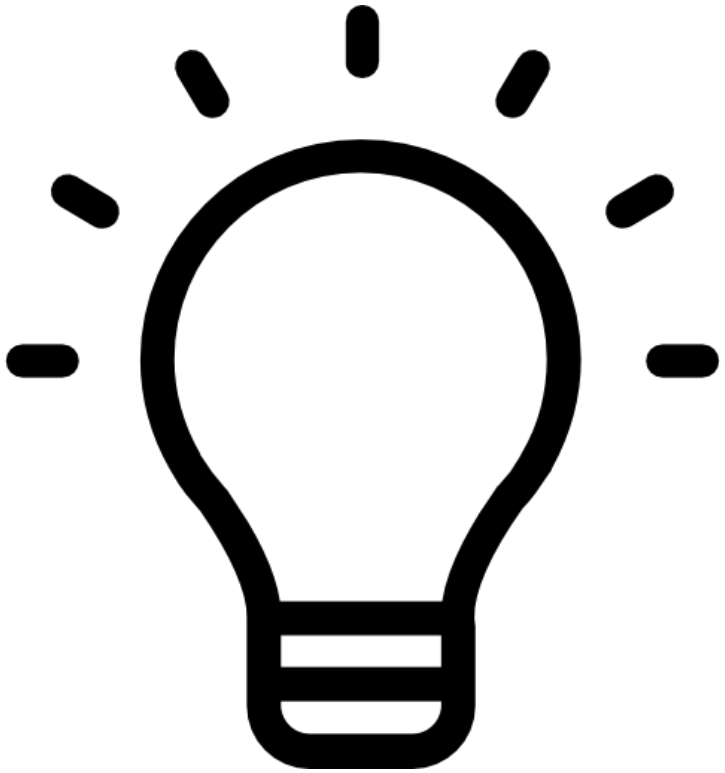


RQ2: which are the main variables between execution time, resource usage and bytecode instruction, that are responsible of the increase in energy consume?

~~time~~ ~~resources~~ bytecode

Main indicators of a bad energy management are useless Object instantiations and a large number of method calls

Future works



- An assessment on the impact that specific bytecode operations has on the overall consumption may reveal best programming techniques to achieve maximum energy optimization
- An analysis of energy consumption in relation to Fowler's code smells, do they bring an improvement or a decrease?
- An analysis on the possible trade-off between green and performance Smells

Grazie per l’attenzione

Identified patterns

section	method identifier
Expensive Calls in Loops	callPropertiesThatCloneValuesInLoop; indirectExceptionHandlingInsideLoops; indirectStringConcatenationInLoops; instantiationInsideLoop; stringConcatenationInLoop; terminationExpressionInLoop;
DB calls	closeDBResourcesASAP; usingLONGDatatype; nullableColumn; selectAll; implicitConversionInWHERE; preferUNIONALL; usingOldStyleJoin; mixAnsiWithOracleSyntax; existsIndipendentClause; withoutPrimaryKey; useDedicatedStoredProcedure; sqlQueriesInsideLoop; tooManyIndexes; firstIndexInWhere; redundantIndexes;
Expensive Object Instantiation	dynamicInstantiation; instantiatingBoolean; largeNumberOfStringConcat; stringInitializationWithObject; declareStaticMethod; boxingImmediatelyUnboxed; numberCtor; randomUsedOnlyOnce; needlessInstantiation; boxedPrimitiveToString; newObjectForGetClass;
Garbage Collector calls	callSuperFinalizeInFinally; callingFinalize;
Poor Programming Choices	arrayCopy; collapsibleIfStatement; deadLocalStoreReturn; formatStringNewLine; localDoubleAssignment; mutualExclusionOR; nonShortCircuit; repeatedConditionals; selfAssignment; staticFieldCollection; switchRedundantAssignment; useArraysAsList; uselessControlFlow; uselessSubstring; usingHashtable; usingRemoveAllToClearCollection; usingStringEmptyForStringTest; usingVector; variableDeclaredAsObject;

Energy results - 1

method name	mean w	mean w/o	impact	p-val	rank
callPropertiesThatCloneValuesInLoop	**15.174**	**4.121**	**-114.568**	**6.86e-18**	**9**
indirectExceptionHandlingInsideLoops	**31.309**	**6.529**	**-130.980**	**6.86e-18**	**9**
indirectStringConcatenationInLoops	**5.907**	**3.295**	**-56.775**	**6.86e-18**	**9**
instantiationInsideLoop	**9.705**	**7.734**	**-22.608**	**6.86e-18**	**4**
stringConcatenationInLoop	**2.321**	**2.065**	**-11.693**	**6.86e-18**	**7**
terminationExpressionInLoop	3.965	3.913	-1.321	2.29e-04	
boxedPrimitiveToString	**4.981**	**3.497**	**-35.003**	**6.86e-18**	**7**
boxingImmediatelyUnboxed	**0.350**	**0.284**	**-20.863**	**4.54e-13**	**4**
declareStaticMethod	**0.370**	**0.327**	**-12.357**	**6.86e-18**	**4**
dynamicInstantiation	**1.139**	**0.343**	**-107.529**	**6.86e-18**	**9**
instantiatingBoolean	**0.356**	**0.295**	**-18.723**	**6.34e-15**	**4**
largeNumberOfStringConcat	1.170	1.819	43.412	6.86e-18	
needlessInstantiation	**0.300**	**0.284**	**-5.442**	**1.67e-02**	**2**
newObjectForGetClass	**0.315**	**0.254**	**-21.349**	**1.59e-14**	**2**
numberCtor	**0.352**	**0.310**	**-12.733**	**1.01e-07**	**4**
randomUsedOnlyOnce	**46.081**	**11.575**	**-119.694**	**6.86e-18**	**9**
stringInitializationWithObject	**0.381**	**0.297**	**-24.882**	**1.05e-15**	**4**
callSuperFinalizeInFinally	11.385	11.427	0.371	6.86e-18	
callingFinalize	16.679	16.769	0.535	6.86e-18	

Energy results - 2

arrayCopy	6.552	6.350	-3.123	8.71e-14	
collapsibleIfStatement	0.295	0.282	-4.669	1.31e-01	
deadLocalStoreReturn	0.274	0.267	-2.696	4.76e-01	
formatStringNewLine	101.758	102.519	0.745	1.95e-01	
localDoubleAssignment	0.309	0.319	3.300	5.84e-02	
mutualExclusionOR	0.256	0.262	2.351	3.45e-01	
nonShortCircuit	**0.353**	**0.269**	**-26.964**	**8.73e-06**	**4**
repeatedConditionals	0.258	0.261	0.892	6.62e-01	
selfAssignment	0.269	0.260	-3.065	2.02e-01	
staticFieldCollection	1.283	1.801	33.584	6.86e-18	
switchRedundantAssignment	0.265	0.268	0.800	8.47e-01	
useArraysAsList	**6.906**	**0.859**	**-155.739**	**6.86e-18**	**9**
uselessControlFlow	0.268	0.262	-2.367	2.82e-01	
uselessSubstring	0.315	0.300	-4.746	1.13e-01	
usingHashtable	**18.952**	**10.651**	**-56.087**	**6.86e-18**	**9**
usingRemoveAllToClearCollection	**6.622**	**4.580**	**-36.459**	**6.86e-18**	**7**
usingStringEmptyForStringTest	0.315	0.309	-1.863	3.33e-01	
usingVector	5.177	5.287	2.105	6.38e-05	
variableDeclaredAsObject	2.916	2.902	-0.512	2.44e-01	

Energy results - 3

method name	mean w	mean w/o	impact	p-value	rank
closeDBResourcesASAP_MYSQL	3.81	3.74	-1.8177	6.77e-02	
existsIndipendentClause_MYSQL	**6.25**	**3.58**	**-54.1686**	**6.86e-18**	**9**
firstIndexInWhere_MYSQL	14.19	14.14	-0.4009	1.08e-01	
implicitConversionInWHERE_MYSQL	3.81	3.79	-0.5157	1.97e-01	
mixAnsiWithOracleSyntax_MYSQL	4.34	4.32	-0.4951	1.32e-01	
nullableColumn_MYSQL	3.69	3.70	0.2052	5.84e-01	
preferUNIONALL_MYSQL	3.67	3.65	-0.6397	1.25e-01	
redundantIndexes_MYSQL	13.82	13.85	0.1856	5.13e-01	
selectAll_MYSQL	3.83	81.26	182.0021	6.25e-01	
sqlQueriesInsideLoop_MYSQL	**17.08**	**3.99**	**-124.3019**	**6.86e-18**	**9**
tooManyIndexes_MYSQL	25.45	24.58	-3.4868	4.00e-01	
useDedicatedStoredProcedure_MYSQL	5.99	6.07	1.2112	3.23e-10	
usingLONGDatatype_MYSQL	4.47	4.47	-0.1997	8.04e-01	
usingOldStyleJoin_MYSQL	4.08	4.08	0.0838	7.41e-01	
withoutPrimaryKey_MYSQL	3.57	75.99	182.0351	4.18e-01	
closeDBResourcesASAP_POSTGRE	2.06	2.04	-1.1264	5.53e-01	
existsIndipendentClause_POSTGRE	**4.47**	**1.98**	**-77.3015**	**6.86e-18**	**9**
firstIndexInWhere_POSTGRE	12.24	12.18	-0.4994	2.25e-01	
implicitConversionInWHERE_POSTGRE	2.08	2.15	3.2811	2.95e-05	
mixAnsiWithOracleSyntax_POSTGRE	1.96	1.96	-0.0921	8.04e-01	
nullableColumn_POSTGRE	2.01	1.99	-1.2010	1.46e-01	
preferUNIONALL_POSTGRE	2.00	1.97	-1.8493	9.93e-03	
redundantIndexes_POSTGRE	**264.87**	**12.61**	**-181.8202**	**6.34e-01**	
selectAll_POSTGRE	2.22	2.23	0.3657	2.58e-01	
sqlQueriesInsideLoop_POSTGRE	**5.37**	**1.99**	**-91.9668**	**6.86e-18**	**9**
tooManyIndexes_POSTGRE	5.43	5.36	-1.3439	5.46e-04	
useDedicatedStoredProcedure_POSTGRE	**2.52**	**2.07**	**-19.4555**	**6.86e-18**	
usingLONGDatatype_POSTGRE	3.13	3.97	23.4586	6.86e-18	
usingOldStyleJoin_POSTGRE	1.94	1.93	-0.5529	4.16e-01	
withoutPrimaryKey_POSTGRE	1.88	1.89	0.5216	3.24e-01	