Static and Dynamic Green Semantic Model for Software

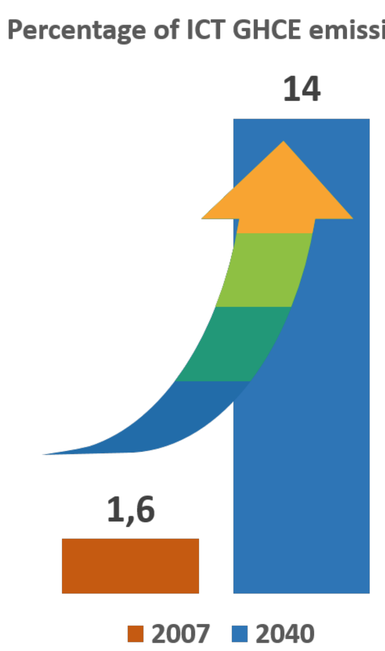
Adel Noureddine

Static and Dynamic Green Semantic Model for Software

Adel Noureddine

Université de Pau et des Pays de l'Adour, LIUPPA, Anglet, France
adel.noureddine@univ-pau.fr

1. Context



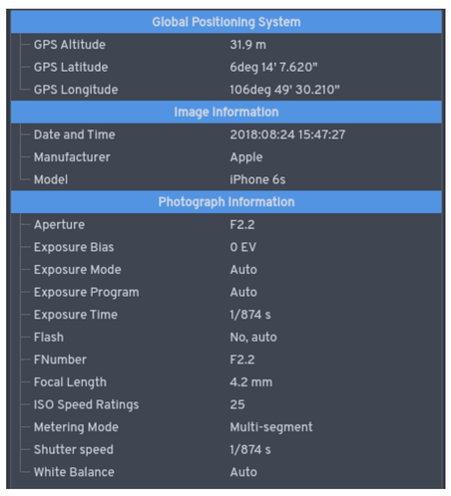
- ICT energy demands and their green house gaz emissions (GHCE) are rising exponentially [1, 2],
- Computers and data centers accounts for the majority of ICT energy consumption [3],
- Most current approaches address energy for specific use cases or individual system layers,
- There is a need to address energy holistically in computing systems.

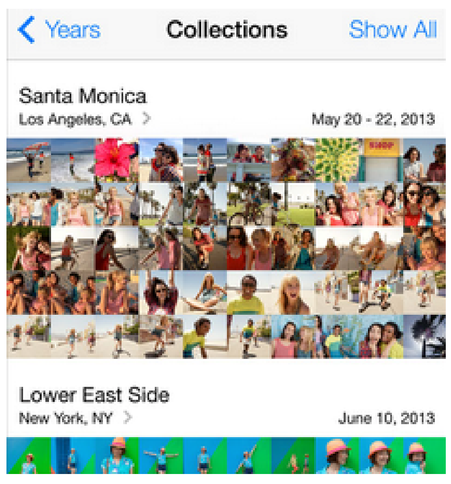
2. Problem Statement

- Huge and heterogeneous data are shared in smart environments (data centers, IoT, software),
- Software specifications (quality, performance) are rarely used in runtime adaptations.

3. What if?

Use EXIF-style metadata for software?





4. Proposition and Goals

Attach a textual model to software, with static quality description of software, and dynamic performance predictions:

- Semantic model to group static and dynamic software metrics,
- Performance and energy prediction models for software,
- Comprehensive software semantic model to be used on runtime for autonomous adaptations.

References

[1] Lotfi Belkhir and Ahmed Elmeligi. Assessing ICT global emissions footprint: Trends to 2040 & recommendations. *Journal of Cleaner Production*, 2018.

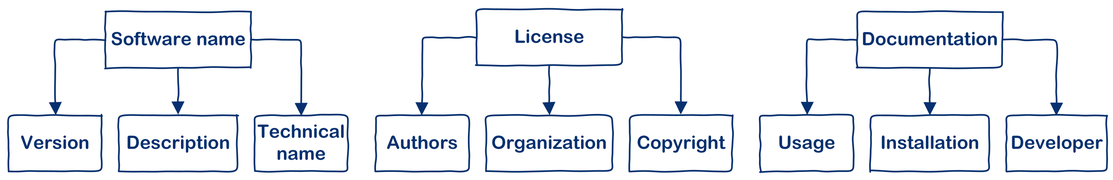
[2] Willem Vereecken, Ward Van Heddeghem, Didier Colle, Mario Pickavet, and Piet Demeester. Overall ICT footprint and green communication technologies. In *Final Program and Abstract Book - 4th International Symposium on Communications, Control, and Signal Processing, ISCCSP 2010*, 2010.

[3] EU FP7 project-288021. Network of Excellence in Internet Science. D8.1 Overview of ICT energy consumption. 2013.

5. Green Software Semantic Model

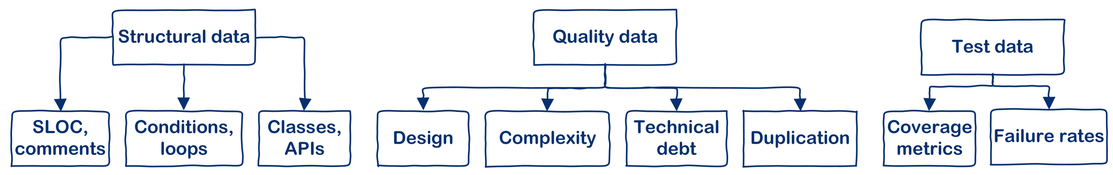
Our green software semantic model is composed of four main categories:

5.1 General metadata



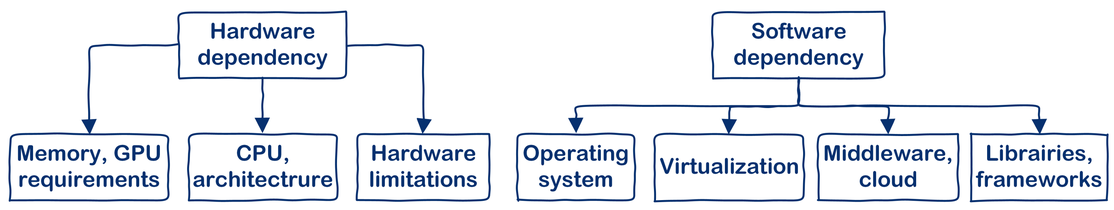
- Metadata about software, its authors, license information and copyright, or documentation,
- Data collected from source code, readme and configuration files (pom.xml, package.json, Makefile).

5.2 Software Quality Information



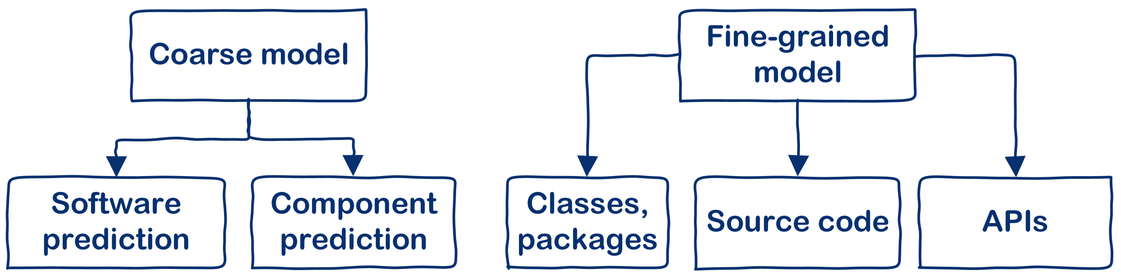
- Code quality and static software metrics,
- Gathered from source code using software analyzers (Frama-C, SonarQube),
- Structural data: metrics describing how the source code is written (SLOC, conditions, APIs),
- Quality data: how well the code is written (code complexity, duplication, technical debt),
- Test data: how well the code has been tested and covered (unit tests, coverage metrics).

5.3 Software and Hardware Dependency Information



- Dependencies upon the software/hardware ecosystem,
- Collected from readme and configuration files, and software analyzers,
- Software dependency such as libraries or frameworks, operating systems, or virtual environments,
- Hardware dependency such as material architecture, recommended or minimum requirements.

5.4 Performance and Energy Prediction Data Models



- Empirical models built using statistical and regression analysis on performance and energy traces,
- Two types of models: a coarse model for higher level components' predictions, and a fine-grained model for source code and APIs' predictions.