Master Internship — 5 to 6 months, starting between Feb. and Apr. 2022

Toward Energy-efficient Software for the Internet of Things

Denisse Muñante (1) and Sophie Chabridon (2)

(1) SAMOVAR Lab, ENSIIE
(2) SAMOVAR Lab, Télécom SudParis, Institut Polytechnique de Paris
Évry, France


Keywords Energy-efficiency, Software engineering, Internet of Things, Distributed systems, Middleware.

Context. Energy efficiency has already been considered for many years at the hardware level. However, powerful and cheaper computing resources have led to less resource optimization in software. Considering the planet resource limits and the increasing role of software in the Internet of things, there is an urgent need to design software with energy awareness as a requirement. In practice, this means to consider energy-efficiency in software quality attributes at design time and then to implement architectural tactics enforcing them. Moreover, usage conditions varying a lot at run-time, dynamic reconfiguration would enable additional energy savings.

Calero and Piattini explore the environmental dimension of software engineering and define Green IT as the sustainable consumption of natural resources by hardware and software components of ICT [1]. Green IT distinguishes between 'Green by IT' and 'Green in IT'. 'Green by IT' considers ICT as a means to optimize the consumption of natural resources (e.g. Smart Home,...), while 'Green in IT' intends to lower the impact of ICT on natural resources. Consequently, Green in Software Engineering corresponds to the subpart of 'Green in IT' which is focusing on methodologies and mechanisms to limit the environmental impact of software products.

Internship objectives. In the context of this internship, a first objective is to take into account environmental sustainability quality attributes, e.g., saving energy consumption or keeping resources consumption low, in the design of software products and to provide recommendations to software engineers for developing software with low environmental impact. In addition to design time, such quality attributes should also be considered at run-time when usage conditions are established. A second objective is to explore dynamic reconfiguration solutions able to monitor energy consumption and the current execution context with the purpose of detecting and alleviating undesired scenarios, e.g., rebound effects, at run-time.

Concretely the tasks that will be carried by the selected candidate:

1. Study the state-of-the-art to collect the metrics [3] used to assess green quality attributes for software products, and to explore the current green architectures used in software engineering life-cycle [6, 2, 4, 7].

2. Artefacts used at design-time: conceive a solution that supports architects and developers to include green quality attributes in the software development life-cycle, and select software architectures by recommending the architectures that respect the green quality attributes:
   — the solution should allow to analyse the dependencies or conflicts among green quality attributes and other software quality attributes and requirements.
   — the solution should consider variation points in software architectures thus allowing dynamic reconfiguration.
   — the solution should contain appropriate (quantitative or qualitative) information that allows the trade-off analysis of the software quality attributes.

3. Artefacts used at run-time: conceive a prototype that allows to explore and select “appropriate” architectures configurations when usage conditions are established and differs from the expected ones, e.g., the actual data flow frequency is not as was expected or a resource state (longevity) make system risky.
   — the solution could be formulated as a multi-criteria problem for the self-reconfiguration decision making process.
   — explore the state-of-the-art methods to solve the multi-criteria decision-making problem for self-reconfiguration, for instance a optimisation algorithm that explores solutions taking into account quality attributes is introduced in [5].
   — implement a prototype to simulate the self-reconfiguration when green quality attributes become deprecated or are violated.
Références


