Software and eco-design

G.EN.ESI Education Centre

'Software and eco-design' is the eighth of an eight part series from the G.EN.ESI Education Centre. The other titles in this series include Life Cycle Thinking, Introduction to Eco-design, Eco-design and Business, Life Cycle Assessment, Legislation and Regulation and Eco-design Case Studies. To read or download any of the titles in this series please visit www.genesi-fp7.eu/education-centre.

Software influence on environmental performance

Whilst software is not tangible, it has environmental effects. Software alone cannot "do" things: it has to be run on a device. This device, whether it is a computer or a dishwasher, has an impact on the environment. Part of this impact can be attributed to software parameters.

The most obvious relationship between software and environmental efficiency is its influence on product energy consumption during use phase. This influence can be positive or negative. The positive influence is where software implementation supports functionalities for low energy consumption (such as automatic stand-by). Negative influence comes from over solicitation of hardware with few functional outputs to user. For example, refreshing the screen every given time, even if the user is not interacting with the product, increases the energy consumption of the screen with no benefits to the user.

The other influence of software is on hardware durability. As is the case for energy efficiency, this influence can be negative or positive. The negative side is the enforcement of software solutions to promote planned obsolescence, deliberately making the hardware stop, or slowing down a function are two of the strategies used in planned obsolescence. But software also provides an opportunity to enhance product durability by making it adaptable to user needs. The integration of new product features through software updates can delay the decommissioning of a product by the user. A good example is firmware update. By providing regular updates of the embedded software of a product, the speed of task realisation can be increased and new features can be provided to the user.

From firmware to applications software, coding has a high potential for improving the environmental impact of the product they are associated to.

Software and Energy Efficiency improvement

Metrics

To improve performance, it is essential to measure it first. The difficulty with software energy efficiency is that the only measurable parameter is the product energy consumption which represents the influence of both hardware and software.

Attributing part of this consumption to software is a difficult task. A proposal made by Synergico[i] is to measure software contribution to energy consumption as a percentage of solicitation of the maximum power requirement for the hardware. Another way to measure this performance is to measure efficiency, i.e. a ratio that represents how close the actual is to a desired situation. Capra et al[ii] proposed the evaluation of the performance of software by testing the energy consumption of different version of the code software on the same hardware.

Since no standard tests are available for software energy efficiency, you can custom your performance indicator depending on your activities:

- Defining a baseline and trying to "beat" it by improving the code tested on a standard hardware, to improve the software performance, or
- Attributing relative energy consumption to software and hardware to be able to work on both parameters if necessary.

Guidelines

A lot of guidelines in energy efficiency need to be supported by software functionalities:

1. Providing live information to the user requires the coordination of a screen with a sensor.
2. Providing the user with adaptable parameters (through simple controls or an interactive interface).
3. Limiting the time in stand-by mode or network stand-by using automation.

This project is co-financed by the European Commission and made possible within the VII Framework Programme

4. Dynamically programming components to turn them on and off depending on the task to be performed.
5. Avoiding oversizing of hardware power by tailoring hardware power capacity to software needs…

Other guidelines are aimed at improving the software itself:

1. Improving ergonomics of the user interface (to perform the task more quickly)
2. Questioning the requirements needs. Not all functions may be worth implementing from the user perspective.
3. Using compact protocol for communication.
4. Considering the energy impact of updates to software…

Software and Durability

Product obsolescence is a major issue for some goods in the electronic sector. A product is considered obsolete when the product is no longer usable by the end-users. The reasons may include:

- The user is not satisfied by the service provided by the product and stops using it.
- Technical issues make the use of the product very difficult to the user and the product is "not working" properly according to the users' standards. If these technical issues occur due to design shortcomings it can be classified as planned obsolescence.

Software can play a role in the first case by improving user satisfaction with the same product. Software updates are aimed at improving product user experience and this can delay the decommissioning of the product.
This is a common solution in application software but it is also of interest for firmware.

Conclusion

Even though eco-design has long been important when designing hardware; software can play a crucial role in improving product environmental impact. It is especially applicable to:

- Improving shortcomings in hardware design by operating the product at lower power,
- Selecting the appropriate hardware for maximum software efficiency,
- Extending product effective lifetime by improving the user experience.

Further Reading

- ECMA, Environmental Design Considerations for ICT & CE Products - ECMA-341. 2008.
- Intel guide to efficient programming:
  https://software.intel.com/sites/default/files/m/d/4/1/d/8/creating_energy-efficient_software.pdf

References

[i] L. Domingo, F. Mathieux, J. Bonvoisin, and D. Brissaud, 'Indicator for in Use Energy Consumption (IUE): a tool enhancing Design for Energy Efficiency of products', presented at the IDMME - Virtual Concept 2010, Bordeaux, 2010.

[ii] E. Capra, C. Francalanci, and S. A. Slaughter, 'Is software "green"? Application development environments and energy efficiency in open source applications', Information and Software Technology, vol. 54, no. 1, pp. 60–71, Jan. 2012.

C. Boks, 'The soft side of ecodesign', Journal of Cleaner Production, vol. 14, no. 15–16, pp. 1346–1356, 2006.

This project is co-financed by the European Commission and made possible within the VII Framework Programme