

One often overlooked factor required for reliable and available systems is that of *power* or *energy*. This factor could be overlooked in the past where most communicating devices were stationary and electrically plugged machines. However, it cannot be ignored in the current networking environment that becomes context- and location-aware, (Yan & Sere, 2004; Roman, Julien, & Payton, 2004), hence it involves mobile, unplugged devices functioning on batteries. These mobile devices have numerous user-friendly features (Michael, 2005) such as high resolution display, radio, camera, TV, GPS, etc. At the same time, a device should be as small as possible, leaving not much space for the battery in spite of the increased energy need of its various features.

We can regard energy in relation to smart devices in a multitude of ways. One aspect is that of being provided by a battery, hence the engineering challenge of fitting a (rechargeable) battery supporting the functionality of the device and thus enabling mobility. Based on this aspect, users can question how long can a certain battery-enabled device be used and how flexible this usage is. For instance, if an individual travels to a remote location and there takes a very special trip, this person may need for the trip only the camera and the phone functionality of a smart device. The rest of the features - software and hardware - should be turned off to save the battery. The current smart devices are sometimes so complex that they do not offer such simple choices to the user. Instead, they may be running some anti-virus software periodically, checking for updates, or loading and keeping a multimedia player ready to be used without a clear choice for the user on whether these features are needed or not. Yet another aspect concerning energy in relation to smart devices is that of deploying or developing products that consume no more than a certain amount of energy per time unit. Recharging a battery is another fundamental issue to consider as well. A clear implication of these energy-related aspects is that we need to take energy into consideration when developing and deploying smart devices: they need to be *energy-aware*.

We can view energy as a non-functional property of a system, and thus not needing to be addressed in the application developer's requirements. Instead, energy could be taken care of by a *network manager*, at a more specialized level often referred to as the *middleware level* (Petre, 2008). Energy consumption would thus be handled together with other properties concerning the proper functioning of smart devices, such as system security or resource availability. However, if we need to specify constraints on a system's overall energy consumption, we have to be able to model it early in the lifecycle, namely starting from the user requirements. This amounts to expressing both functional and non-functional properties at the same abstraction level, thus breaking some of the traditional encapsulation of the hardware and middleware specification. We claim that this is a necessary modeling choice for energy-awareness as well as beneficial for the efficiency of the developed systems and adhere to it in this paper. Energy becomes a feature of every single module, software or hardware, thus enabling a wide range of properties to be proved for the developed device.

Another key dimension in computing today is that of *sustainability*. In a strict sense, this refers to using resources without depleting them, thus falling along the power consumption and efficiency lines of research. However, as our society becomes more computing-dependent, we have become proficient in developing most various types and forms of software-based systems. The time is therefore ripe for trying to develop software-based systems in a *sustainable manner*, at various levels: less human resources and less time involved in planning, developing, and testing; less redundant and useless software; no incorrect development of requirements into software. These objectives can be approached by investing in the devise of more sophisticated methods and tools that take care of avoiding the undesired side-effects of developing software.