Towards Automatic SW Integration in Dependable Embedded Systems

Leandro Batista Ribeiro and Marcel Baunach, TU Graz

Embedded systems must support regular software updates, due to changes in legal regulations, improved algorithms, bug/security fixes, etc. Currently, most embedded systems only support updates through full image replacement. This image is a monolithic software statically built,i.e., all pieces of software are integrated at build time. This approach has several disadvantages, such as the need to rebuild the full software even when only a small portion is modified, and the need to stop the functionality and reboot the device upon updates, which is unacceptable on many (safety-)critical applications, e.g., in medical or nuclear fields. In this paper, we present the modular architecture used in MCSmartOS, and how it supports dynamic updates at module level. This is a first step to enable automatic integration on embedded devices, i.e, to receive a piece of software and incorporate it into the running system. However, many embedded devices are classified as "dependable systems", which must satisfy a set of functional and non-functional properties (FPs/NFPs) w.r.t. real-time, safety, security, and maintainability. Thus, before integrating a new module into the software stack, it is necessary to ensure that the target device will still satisfy all required properties after the update. Therefore, the automatic integration must include a pre-validation, called Compatibility Check (CC). The CC performs various operations: from simple ones, such as checking if a device has enough memory to store the new software, to complex NFP checks, such as schedulability analysis and deadlock detection. Due to the wide range of concepts involved in the CC, it is unfeasible to cover it in this work. It will be instead discussed in future publications. Some of the CC's operations require so much performance or memory that many embedded devices cannot afford to execute them. Hence, we also present a client-server update protocol that distributes update operations between the embedded devices (clients) and high-performance servers that provide the updates. This way, we are able to dynamically update embedded devices of different performance classes: the more resource-constrained a device is, the more operations it outsources to the server.