4.1.1 Platforms

Return to Functional Requirements

A **Platform** is an overloaded term and depends on the context it is used in. Sometimes, Platform refers to just the hardware (i.e., x86, 68000, CISC, RISC, ARM, etc.), other times it can refer to the Operating system (i.e., Windows, Linux, MacOS, Android, iOS), sometimes it can refer to the run-time environment provided by the programming languages used (i.e., C, C++, C#, Java or .NET), while othertimes it can refer to the networking used to connect computers together (i.e., Transmission Control Protocol (TCP)/Internet Protocol (IP)/User Datagram Protocol (UDP), Bluetooth, ZigBee).

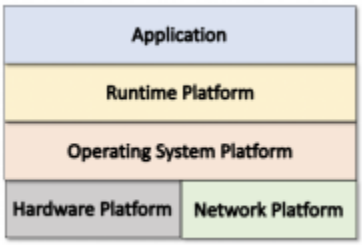

Figure 1: The Kinds of Platforms

As a consequence, in order to identify a specific platform, all the individual platforms and the specific versions need to be identified. This could result in hundreds if not thousands of combinations of platforms and platforms. This can quickly become a maintenance nightmare with versions that can almost change daily to apply patches.

```
HW-vvv:OS-vvv:RT-vvv
```

Where:

- HW - represents the specific hardware such as x86, 6800, ARM, etc
- OS - represents the specific Operating System such as Windows, MacOS, iOS, Android, etc
- RT - represents the specific runtime environment such as C, Java, Solidity, C#, eyc
- vvv - represents the specific version of the platform such as 10.15.7

Another way to represent a platform is to use an Application Container that encapsulates the platforms into the container. This simplifies the deployment and the number of user platforms that have to be supported. For example, any user platform that can support a container, will be able to deploy and use the Application.

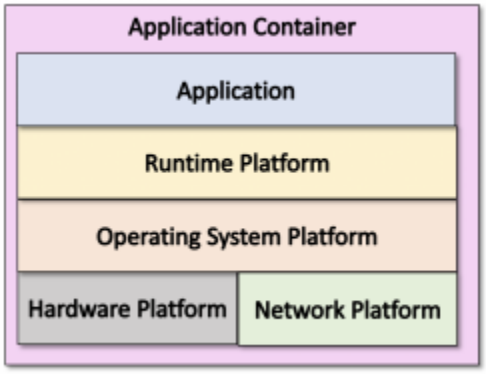

Figure 2: The composition of a Application Container.

- 4.2.1.5 Virtualized Nodes