Chapter 60
Tools and Platforms for Developing IoT Systems

Görkem Giray
https://orcid.org/0000-0002-7023-9469
Independent Researcher, Turkey

ABSTRACT

The internet of things (IoT) transforms the world in many ways. It combines many types of hardware and software with a variety of communication technologies to enable the development of innovative applications. A typical IoT system consists of IoT device, IoT gateway, IoT platform, and IoT application. Developing these elements and delivering an IoT system for fulfilling business requirements encompasses many activities to be executed and is not straightforward. To expedite these activities, some major vendors provide software development kits (SDK), integrated development environments (IDE), and utility tools for developing software to be executed on IoT devices/gateways. Moreover, these vendors utilize their cloud platforms to provide fundamental services, such as data storage, analytics, stream processing, for developing IoT systems. These vendors also developed IoT specific cloud-based services, such as connectivity and device management, to support IoT system development. This chapter presents an overview of tools and platforms provided by five major vendors.

INTRODUCTION

The Internet of Things (IoT) has attracted considerable interest in many various domains, including smart cities, retail, logistics, manufacturing, agriculture, and health. In line with this interest, designing and developing IoT systems have become mainstream. In theory, it is possible to develop an IoT system using many types of hardware, developing software with many tools and programming languages and even designing a special communication protocol for data exchange. On the other hand, using standard hardware and communication protocols, reusing software libraries and platforms have many advantages, including: (1) decrease in development time and hence time-to-market; (2) decrease in cost due to economies of scale; (3) interoperability with other systems via standards. This chapter attempts to provide an

DOI: 10.4018/978-1-6684-3702-5.ch060

overview of some tools and platforms that can be used for developing software for the IoT. These tools encompass Software Development Kits (SDK), Integrated Development Environments (IDEs), utility tools for developers, and IoT platforms that provide some common basic functionalities for IoT systems.

Section 2 begins with an overview of the foundational concepts underlying the IoT and then presents the brief descriptions of the elements making up a typical IoT system. Section 3 summarizes the tools and platforms offered by major vendors (Amazon, Bosch, Google, IBM, and Microsoft) to develop IoT systems. Section 4 discusses the current state of software development for IoT devices, the fundamental services delivered by IoT platforms and capabilities offered by IoT platforms for handling IoT big data. Finally, Section 5 concludes the chapter.

BACKGROUND

The "IoT" concept and IoT systems make use of many concepts that are available in various disciplines, including software engineering, software architecture, and cloud computing. The following subsection titled "foundational concepts" summarizes these concepts and the relationships among them. The second subsection presents the main components of a typical IoT system.

Foundational Concepts

IoT systems are made up of many distributed components interacting via a network. To cope with the complexity of such a system, *abstraction* and *encapsulation* are key concepts. A group of functionality can be encapsulated and provided as services; which is the approach of *Service Oriented Architecture (SOA)*. *Cloud computing* extends the scope of these services to infrastructures, platforms, even applications. The remainder of this section briefly describes these concepts and clarifies the relationships among these concepts.

Abstraction is one of the fundamental cognitive activity associated with problem-solving. Abstraction is a technique for coping with complexity. Abstraction means preserving essential and eliminating unessential information about an entity from a specific perspective for an objective. This helps to focus on the big picture and thus cope with the complexity. Encapsulation complements abstraction by hiding the internal functioning of an entity and providing an interface to exhibit the behavior of the entity. In short, an entity is known by its interface presenting its services to outside world and its internals is hidden from the outside world. The examples of such an entity vary in size and can be a procedure, a class, a layer, a library, even a platform. For instance, encapsulation in object-orientation is the packaging of attributes representing state and operations into a class and provide an interface to access the services provided by that class (Page-Jones, 1999). Figure 1 illustrates how an IoT device can be abstracted and represented as a class. Data and implementation details are encapsulated in the class; the operations of sending data and resetting are provided to the outside world.

Abstraction and encapsulation can be done at any level, at a micro level, such as a class, at a higher level, such as establishing layers while designing the architecture of a system. An architecture encompasses the major components and their interrelationships at a certain level of detail in line with the objective in drawing that architecture. Figure 2 illustrates how the components of an IoT system can be organized as layers to cope with complexity (Köksal & Tekinerdogan, 2017). Each layer has a group of cohesive responsibilities, such as security layer providing the security functionality for the system. Each layer