SCIENCE CHINA
Information Sciences

• LETTER •

May 2023, Vol. 66 159101:1–159101:3
https://doi.org/10.1007/s11432-020-3213-2

# A software-defined MAPE-K architecture for unmanned systems

Mingyue JIANG[1], Libin ZHENG[1], Zuohua DING[1*] & Zhi JIN[2]

[1]*School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou* 310018*, China;*
[2]*Key Laboratory of High Confidence of Software Technologies (MoE), Peking University, Beijing* 100871*, China*

Received 15 October 2020/Accepted 15 December 2020/Published online 25 November 2022

**Citation** Jiang M Y, Zheng L B, Ding Z H, et al. A software-defined MAPE-K architecture for unmanned systems. Sci China Inf Sci, 2023, 66(5): 159101, https://doi.org/10.1007/s11432-020-3213-2

Dear editor,

With the advances in software, sensors, and communication technologies, there is rapid development in unmanned systems. An unmanned system consists of unmanned vehicles, a control system and a communication system among its components. These kinds of systems, due to their specific autonomy and high mobility, offer a wide range of applications and have been used as important tools in various domains. For example, the usage of unmanned aerial vehicles ranges from military applications (such as battlefield inspection) to public, civil, and even personal applications [1, 2].

Various architectures have been proposed by considering different aspects of unmanned systems, such as the network and communications [3], co-operations and traffic exchanges [4]. However, an unmanned system is a complex system, where various different categories of resources are located in a dynamic, evolving environment. For such a system, effectively managing and scheduling resources to fulfill the users' requirements are non-trivial [5]. To the best of our knowledge, none of the existing architectures focuses on the resource management and scheduling of unmanned systems.

The software-defined paradigm is well-known for its high abstraction, effective management, and high flexibility [6, 7]. Moreover, it has gained great success in several application domains, such as networking, storage, and smart home. In this study, we propose to apply the software-defined principles to unmanned systems, and present a software-defined architecture for unmanned systems by adapting the MAPE-K structure [8].

The architecture mainly consists of three layers: the application layer, the controller layer, and the device layer. The application layer supports various applications of the unmanned systems, and the device layer consists of various hardware devices (such as the camera and different types of sensors) of the unmanned system. The controller layer mainly contains a controller that is responsible for central management and a resource pool that provides information for the system.

Basically, the controller layer is responsible for resource management and scheduling. The former refers to the maintaining and managing of all resources of the system (it is expected that both the software and hardware resources are uniformly managed in a software-defined way), while the latter refers to the procedure of collecting the users' requirements and the environmental information, checking the system's status, and then re-configuring the system to form a feasible and effective work flow to fulfill the users' requirements or to adapt the system to a new environment. Therefore, we propose to adapt the MAPE-K structure and the software-defined principles to design the controller, the details of which are presented below.

(1) The basic components of the controller. Figure 1(a) presents the basic MAPE-K structure [8], which is composed of four components (that is, the monitor component, the analyze component, the plan component and the execute component) and one knowledge base. The knowledge base, which refers to the resource pool of the controller layer, is used to maintain all information of an unmanned system. Nevertheless, the four components respectively focus on specific tasks. The monitor component is responsible for collecting the users' requirements and the environmental information; the analyze component is used for analyzing the collected information; the plan component focuses on making decisions by referring to the users' requirements, the environmental information and the current status of the system; and the execute component is used for operating on all required resources to satisfy the users' need. To accomplish a given task, the MAPE feedback loop keeps interacting with the knowledge base in order to acquire the current status of the system such that appropriate decisions can be made and the whole system can be managed in a proper way.

(2) Software-defined software services and hardware devices. One key issue of an unmanned system is to properly describe the information of resources such that different resources can be abstracted and managed in a uniform way. To this end, we propose the software-defined software services and hardware devices by following the idea of metamodeling [9]. The reason for adopting metamodeling is that it focuses on concepts and properties associated with certain

* Corresponding author (email: zouhuading@hotmail.com)

 

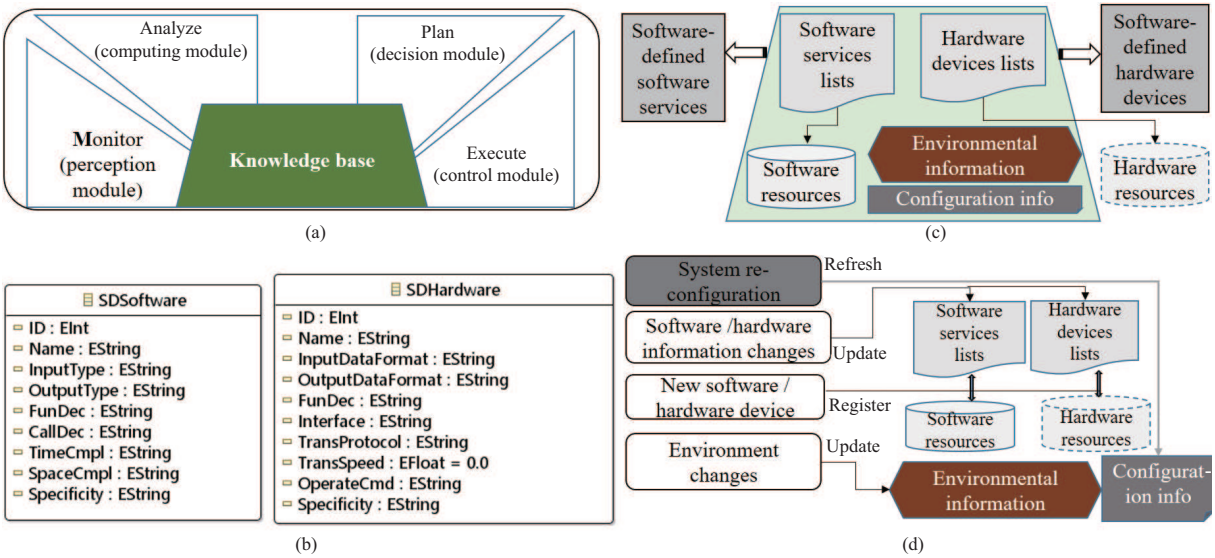


**Figure 1** (Color online) The MAPE-K based controller for unmanned systems. (a) The overview of MAPE-K structure; (b) software-defined software services and hardware devices; (c) the design of the knowledge base; (d) the evolution of the knowledge base.

entities or domains rather than specifying the implementation details, which is consistent with the intuition behind resource abstraction. We define two metamodels, namely, SDSoftware and SDHardware, which respectively specify the way of describing a software service and hardware device. In this way, any software service (or hardware device) is an instance of SDSoftware (or SDHardware).

The details of SDSoftware and SDHardware are shown in Figure 1(b). The information of a software service or a hardware device is abstracted by considering the following four aspects. The first aspect is the basic information, which includes the identifier (attribute ID) and the name (attribute name) of a service or device. The second one is the functionality properties, which are reflected by the functionality description of software services or hardware devices (attribute FunDec), and the algorithm complexity of software services (attributes TimeCmpl and SpaceCmpl) or the performance properties of hardware devices, such as the data transfer speed (attribute TransSpeed). The next aspect focuses on the usage properties, which are reflected via the input and output data types (attributes InputType and OutputType of SDSoftware, and attributes InputDataFormat and OutputDataFormat of SDHardware), and the way of invoking software services (attribute CallDec of SDSoftware) or operating devices (attributes Interface, TransProtocol, and OperateCmd of SDHardware). The last aspect relates to some distinct characteristics of the service or device, which is described by the attribute Specificity.

(3) Knowledge base and its evolution. The knowledge base maintains all information required by the controller to accomplish resource management and scheduling. We design the knowledge base by leveraging the software-defined software services and hardware devices, the overview of which is shown in Figure 1(c). Generally, the knowledge base maintains four types of information: the list of available software services, the list of hardware devices, the environmental information, and the system configuration information. For the former two types of information, they are represented according to the software-defined software services and the software-defined hardware devices (as specified by SDSoftware and SDHardware), respectively. It should be noted that each software service record in the list refers to one of the software services in the software pool, and each hardware device record in the list refers to one of the hardware devices in the unmanned system.

An unmanned system has to deal with dynamically varying requirements and environments. Hence, the knowledge base is expected to keep track of all information of the system in order to reflect the latest status of the system. The evolution of the knowledge base is described in Figure 1(d) and is explained below.

- When some new software services or hardware devices are incorporated into an unmanned system, they need to be registered into the system. To register a software service or a hardware device, the relevant resources are loaded into the system (which are maintained by the software resource pool or the device layer). Then, essential information of the service or device is extracted according to SDSoftware or SDHardware, creating a new instance to form a new element of the software service list or the hardware device list.

- When some existing software services or hardware devices change, the variations will be handled and reflected in the knowledge base such that the information stored in the knowledge base is always consistent with the status of the system resources. This is accomplished by the update of the knowledge base. To update the information of a service or device, the first step is to search for the corresponding element in the software service list or the hardware list, and then the changes are identified, based on which the relevant parts of the element are updated.

- The knowledge base also updates when the environments change. In this case, a similar procedure (as that used for handling resource changes) is conducted to process the environmental information.

- Whenever the system adapts to a new requirement or a new system environment, the system's configuration needs to be refreshed. The refresh is accomplished by first identifying the differences between the old configuration and the new configuration, and then updating the configuration in-

formation stored in the knowledge base.

It is noted that the former three types of operations for updating the knowledge base are all triggered by the notification from the monitor component, while the last type of operation is triggered by the notification from the plan component.

*Experiments.* We built a prototype upon ROS (version: melodic 1.14.6) to implement the proposed architecture. The prototype mainly implements the controller layer and the application layer. It further leverages the Gazebo multi-robot simulator (version: 9.0.0) to simulate unmanned vehicles and environments, and applies the rviz (version: 1.13.12) to graphically display vehicles' behaviors and the environments. We adopted the task assignment as an application scenario, where $n$ vehicles and $m$ destinations ($m \geqslant n$) are given, and each vehicle is configured with an initial destination and then is guided to its destination to accomplish a task. In this scenario, the requirement change refers to adding or removing destinations, while the environmental change is reflected by the positions of vehicles. The experimental results show that when the user alters the requirements, the system is able to capture these changes and then properly re-arrange the system resources to form a work flow for satisfying the users' needs.

*Conclusion.* In order to support the resource management and scheduling of unmanned systems, this study presents a software-defined MAPE-K architecture. The proposed architecture consists of three layers (the application layer, the controller layer and the device layer), among which the control layer is responsible for managing and scheduling resources to fulfill the users' requirements. To this end, the software-defined software services and hardware devices as well as the MAPE-K structure are adapted to design the controller. An experiment built upon a prototype implementing the architecture demonstrated the feasibility of the proposed architecture.

**Acknowledgements** This work was supported by National Natural Science Foundation of China (Grant Nos. 61210004, 61170015, 61751210, 61802349) and Zhejiang Provincial Natural Science Foundation of China (Grant No. LY20F020021).

**References**

1 Verfuss U K, Aniceto A S, Harris D V, et al. A review of unmanned vehicles for the detection and monitoring of marine fauna. Mar Pollution Bull, 2019, 140: 17–29
2 Zeng Y, Zhang R, Lim T J. Wireless communications with unmanned aerial vehicles: opportunities and challenges. IEEE Commun Mag, 2016, 54: 36–42
3 McCoy J, Rawat D B. Software-defined networking for unmanned aerial vehicular networking and security: a survey. Electronics, 2019, 8: 1468
4 Krichen L, Fourati M, Fourati L C. Communication architecture for unmanned aerial vehicle system. In: Ad-hoc, Mobile, and Wireless Networks. Cham: Springer, 2018. 213–225
5 Idries A, Mohamed N, Jawhar I, et al. Challenges of developing UAV applications: a project management view. In: Proceedings of International Conference on Industrial Engineering and Operations Management (IEOM), 2015. 1–10
6 Mei H. Understanding "software-defined" from an OS perspective: technical challenges and research issues. Sci China Inf Sci, 2017, 60: 126101
7 Mei H, Guo Y. Toward ubiquitous operating systems: a software-defined perspective. Computer, 2018, 51: 50–56
8 Kephart J O, Chess D M. The vision of autonomic computing. Computer, 2003, 36: 41–50
9 Atkinson C, Kuhne T. Model-driven development: a metamodeling foundation. IEEE Softw, 2003, 20: 36–41