# A Study of Software Health Management in Unmanned Aerial Vehicle Systems

Jinseok Park[1], Eu-Teum Choi[2] and Seongjin Lee[3,*]

[1]Dept. of Aerospace Software Engineering, Gyeongsang National University, Jinju, Republic of Korea
[2]Convergence Research Center for Materials and Mechanical Systems, Jinju, Republic of Korea
[3]Dept. of AI Convergence Engineering, Gyeongsang National University, Jinju, Republic of Korea

**Abstract**

Unmanned Aerial Vehicles (UAVs) must operate their installed software without errors to complete their mission. Software faults cause errors in the software. Most of them are removed through the Verification and Validation process in the development phase. However, developers and testers cannot eliminate all faults due to the inability to account for all uncertain environments and conditions. System health management is the technology that helps UAVs successfully perform missions by mitigating errors during operation. However, there are errors that the existing UAV System Health Management cannot diagnose because errors are not deterministic nor explicit. This paper analyzes two UAV systems, PX4 and Ardupilot, to categorize software errors. It also assesses whether the existing UAV System Health Management can diagnose these errors. Our study can help developers and researchers better understand the characteristics of software errors in UAV systems and set the direction for future work in UAV software health management.

**Keywords**

Unmanned aerial vehicles, software errors, software health management

## 1. Introduction

Unmanned Aerial Vehicles (UAVs), widely used in modern society, are intricate systems that combine hardware and software components to execute specific functions. For safety-critical systems like UAVs, software errors can potentially lead to catastrophic failures [1], endangering both people and property [2]. Therefore, ensuring the installed software operates without errors is crucial for UAVs to prevent such catastrophic failures and guarantee the safety of both individuals and assets.

Errors are caused by software faults. Most faults are eliminated through significant verification and validation processes during the development phase. However, due to the size and complexity of the software and the presence of uncertain environments and conditions, it is impossible to eliminate all faults. Particularly, for UAVs, which often do not have the obligation to undergo the strict and substantial V&V processes required for manned aircraft according to DO-178C [3], there may be an even more pressing need to manage software faults that are not eliminated by V&V [4].

*ISE 2023: 2nd International Workshop on Intelligent Software Engineering, December 4, 2023, Seoul*
*Corresponding author.
phk7298@gnu.ac.kr (J. Park); etchoi@gnu.ac.kr (E. Choi); insight@gnu.ac.kr (S. Lee)

To ensure safe and reliable UAV operation, UAV systems need to possess health management capabilities. UAV System Health Management (SHM) [4, 5, 6, 7, 8, 9] is a critical technology that detects, diagnoses, predicts, and mitigates errors occurring during operations, thereby helping in the successful execution of UAV missions. Existing UAV SHM primarily focuses on diagnosing errors related to safety requirements in sensors and software. However, actual UAV software errors encompass such a vast range of root causes and symptoms that there are software errors that cannot be diagnosed by the existing UAV SHM.

This paper presents a study on UAV software health management, focusing on actual software errors in UAV systems. Based on studies [10, 11] that analyzed the characteristics of UAV-specific errors documented in issue reports of well-known open-source UAV systems (i.e., PX4 [12] and ArduPilot [13]), as well as studies [14, 15] that analyzed root causes and symptoms of errors in Autonomous Vehicles (AV) and Cyber-Physical Systems (CPS), including UAVs, we classified UAV software errors. Through this classification, we assessed the diagnostic range of existing UAV system health management [4, 6, 7, 8, 9]. The results show the necessity for new work in health management techniques targeting software execution errors in UAVs. In conclusion, this paper's contributions are as follows:

1. We analyze the results of recent research on software errors in UAV systems.
2. We identify the diagnostic range of UAV SHM based on the classified errors.

Section 2 of this paper explains UAV SHM techniques and UAV software errors. Section 3 describes the subjects of error analysis and the classification method. Section 4 presents the main results of our study. Section 5 discusses the implications of our work. Section 6 concludes the paper.

## 2. Background

### 2.1. UAV System Health Management

The goal of UAV System Health Management (SHM) is to prevent failures in UAV system functions during operation. In cases where errors occur in system components, sensors, or flight software during operation, SHM should detect and diagnose these errors before they lead to failures. Subsequently, appropriate mitigation must be taken to eliminate the errors.

Moosbrugger et al. [6], Rozier et al. [7], and Schumann et al. [8, 4] conducted real-time monitoring using FPGA-based arithmetic circuits for UAV sensors and software signals. They based their approach on model-based analysis and temporal logic evaluation, followed by the application of Bayesian networks. This allowed them to diagnose errors related to safety requirements in the UAV system. Zermani et al. [9] conducted real-time monitoring of UAV sensors and software signals based on CPU/FPGA. Through Bayesian networks from Failure Mode and Effects Analysis (FMEA), they diagnosed and predicted errors related to GPS and battery in the UAV system.

### 2.2. UAV Software Errors

UAV software errors occur due to faults and can lead to system failures. UAV software errors vary in their causes and symptoms. They can arise from incorrect source code or unexpected environmental and situational factors. Some errors may only manifest during operation and can be challenging to reproduce. Symptoms also may vary depending on errors or environmental and conditional contexts.

Taylor et al. [10] investigated the root causes, symptoms, and recovery strategies for 277 software errors occurring in PX4 and ArduPilot. Wang et al. [11] examined the root causes, symptoms, and reproducibility of 569 software errors in PX4 and ArduPilot. Garcia et al. [14] and Zampetti et al. [15] classified the root causes and symptoms of software errors occurring in AV and CPS.

## 3. Methodology

### 3.1. Error Classification

We analyzed two UAV systems, PX4 and ArduPilot. These UAV systems are well-known and stable active open-source platforms. These systems have well-maintained error reports, making error analysis easier.

To systematically classify error types, we referred to studies [10, 11] that analyzed the characteristics of UAV software errors in PX4 and ArduPilot, as well as studies [14, 15] that analyzed root causes and symptoms of software errors in AVs and CPS.

### 3.2. UAV System Health Management Analysis

To analyze UAV SHM, we performed the following procedure: (1) Selection of studies related to UAV SHM that address not only hardware-related issues such as sensors but also software errors. (2) Identification of the subsystems and monitoring targets covered by these selected studies. (3) Confirmation of diagnosable software error types. As a result, we determined the range of software error diagnosis within the existing UAV SHM [4, 6, 7, 8, 9].

## 4. Research Questions

In this section, we provide the research findings related to UAV software error classification and SHM diagnosis range. We address the following two research questions (RQs):

- RQ1: What types of errors occur in UAV software?
- RQ2: What types of software errors does UAV SHM diagnose?

By studying RQ1, we classified 10 root causes into three high-level categories. By studying RQ2, we identified types of software errors that the current UAV SHM can diagnose.

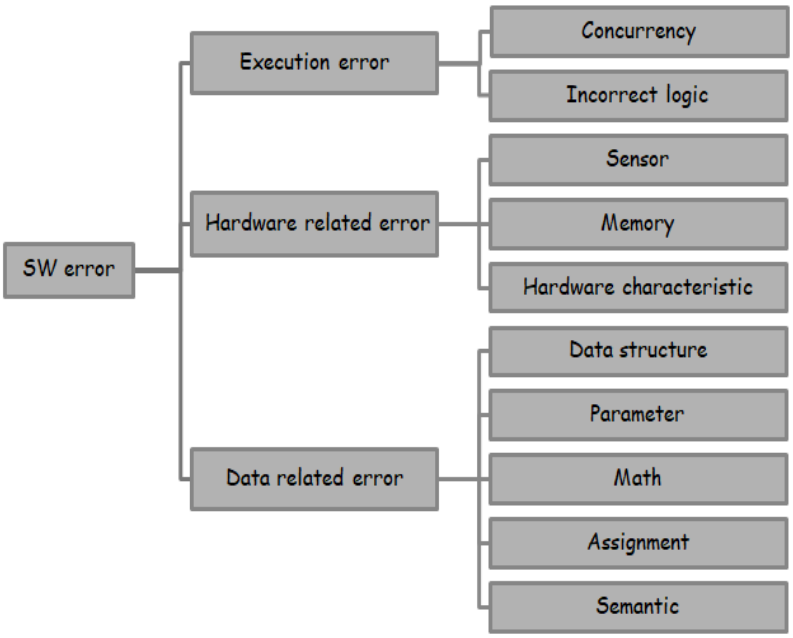


**Figure 1:** The Categorization of UAV Software Errors

### 4.1. RQ1: What types of errors occur in UAV software?

Figure 1 illustrates the root causes of the UAV software errors we classified. We analyzed 10 different root causes and categorized them into three high-level categories:

- Execution error – Errors occurring in the internal execution processes of software components. These errors are not related to sensors or peripheral device data but rather stem from the software flow or logic.
- Hardware-related error – Errors caused by hardware components such as sensor data or peripheral devices and the associated software.
- Data-related error – Errors related to the usage of UAV data, including data structures, allocation, calculations, etc.

We categorized errors based on three criteria. First, errors occurring from the internal execution of software components. These are primarily related to the software itself and have minimal hardware dependencies. Second, hardware-dependent errors. These are influenced by the environment and distinct from other software errors as they are specific to UAVs. Third, errors related to data configuration and processing. These are dependent on the designed data structures and code. The following are descriptions of the root causes for the root causes within these three categories:

1. Root Causes of Execution Error

| UAV SHM | Monitoring Target | | | |
|---|---|---|---|---|
| | GPS System | Battery System | File System | GN&C System |
| [4, 6, 7, 8] | The number of GPS satellites and Position accuracy | Battery voltage and current | Message queue length, Error and available space in file system | All messages from GN&C and flight data |
| [9] | Sensor data and Appearance context | Battery energy, Internal and External events | - | - |

**Table 1**
The Sub-Systems and Monitoring Targets Managed by UAV SHM Studies

- Concurrency: Errors that only occur in multithreaded environments (e.g., data races and deadlocks).
- Incorrect logic: Incorrect implementation of condition logic.

2. Root Causes of Hardware-related Error
   - Sensor: Incorrect use of sensor data (e.g., reading an incorrect altitude value).
   - Memory: Incorrect use of memory (e.g., no available memory space or improper memory allocation).
   - Hardware characteristic: inconsistency or priority issues between hardware and software due to software developer's Misconception of hardware characteristics.
3. Root Causes of Data-related Error
   - Data structure: Incorrect data structure.
   - Parameter: Improper handling of parameters.
   - Math: Incorrect numerical calculations, formulas and values.
   - Assignment: Incorrect assignment or initialization of variables.
   - Semantic: Errors that do not misuse computational machinery, but violate the user's expectations of UAV behavior.

## 4.2. RQ2: What types of software errors does UAV SHM diagnose?

To understand RQ2, we first analyzed the sub-systems and monitoring targets managed by UAV SHM studies [4, 6, 7, 8, 9] (Table 1).

Studies [4, 6, 7, 8] utilizing model-based analysis, temporal logic evaluation, and Bayesian network for real-time managing of UAV sensors and software diagnose errors that violate UAV's flight safety requirements [16]. The following explains the sub-systems and monitoring targets related to flight safety requirements:

| UAV SHM | Diagnosis of Software Error | | |
|---|---|---|---|
| | Execution | Hardware-Related | Data-Related |
| Schumann [4, 8] | X | O | O |
| Moosbrugger [6] | X | O | O |
| Rozier [7] | X | O | O |
| Zermani [9] | X | O | X |

**Table 2**
The Diagnosis Range of UAV SHM Studies

- GPS system - Monitors GPS location accuracy and the number of GPS satellites to diagnose errors based on their correlation.
- Battery system - Monitors battery voltage and current to diagnose any errors in their values.
- File system - Monitors the length of the message queue written in the file system and the remaining space in the file system to diagnose errors related to flight rules.
- GN&C (guidance, navigation, and control) system - Monitors all messages and flight data transmitted by the GN&C system to diagnose compliance with flight rules.

Study [9] that uses Bayesian networks from the FMEA for real-time monitoring of UAV sensors and software diagnoses errors targeting the GPS and battery systems. The following provides explanations of the monitoring targets:

- GPS system - Monitors sensor data and contextual information to diagnose error probabilities.
- Battery system - Monitors activation commands, internal and external errors or events, and battery energy consumption to diagnose error probabilities.

Table 2 summarizes the scope of UAV SHM research in diagnosing software errors based on the error categories of RQ1. The following are the results of RQ2:

1. We found that none of the 5 UAV SHM studies diagnose execution errors. These studies do not consider errors arising from software internal execution, such as concurrency errors or incorrect logic, as monitoring targets, and therefore do not diagnose execution errors.
2. We found that all 5 UAV SHM studies diagnose hardware-related errors. These studies diagnose hardware-related errors by considering sensor data, the relationship between hardware and software, as well as the physical environment in which the system operates.
3. We found that 4 out of the 5 UAV SHM studies diagnose data-related Errors. These studies diagnose data-related Errors by considering factors such as data range and allocation in the way data is used.

## 5. Discussion

Our study can have implications for both UAV system developers and UAV SHM researchers. For developers, our classification of UAV software errors provides insight into issues that require attention in UAV system development. These systems encounter errors related not only to the software itself but also to the hardware. Thus, the characteristics of these UAV system errors demand knowledge about both software and hardware.

For researchers, our study triggers efforts to enhance the diagnostic range and mitigation capabilities of UAV SHM. Existing UAV SHM has mostly focused on software errors related to hardware or data. Execution errors, such as concurrency errors, cannot be eliminated during the development phase due to their non-deterministic characteristics, influenced by the environment or conditions. This indicates the necessity for health management techniques targeting software execution errors.

Finally, since the existing UAV SHM only performs diagnosis and prediction, it should address an error mitigation process. Utilizing diagnostic and predictive information for mitigation functions is essential to prevent system failures.

## 6. Conclusion

UAVs are safety-critical systems that must prevent catastrophic failures due to software errors. Given the nature of software errors, it is not possible to eliminate all errors during the development phase. Therefore, managing errors that occur during operation through UAV software health management is essential.

This paper studied software errors occurring in PX4 and ArduPilot and existing research on UAV SHM. We analyzed ten fundamental causes of UAV-specific software errors and categorized them into three top-level categories. Based on this classification, we examined the error diagnostic scope of UAV SHM and discovered that not all UAV SHMs diagnose errors resulting from incorrect execution within software components, i.e., execution errors. Our study results highlight the need for future studies in software health management techniques targeting UAV software execution errors.

## Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2021R1A2C1014163).

## References

[1] A. N. Srivastava, J. Schumann, Software health management: a necessity for safety critical systems, Innovations in Systems and Software Engineering 9 (2013) 219–233.

[2] L. M. Hilton, Personal injury and property damage with drones, Retrieved Jan 25 (2017) 2021.

[3] S. C. in Airborne Systems, E. Certification, DO-178C, 2011.

[4] J. Schumann, K. Y. Rozier, T. Reinbacher, O. J. Mengshoel, T. Mbaya, C. Ippolito, Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems, International Journal of Prognostics and Health Management 6 (2015).
[5] D. Jing, W. Haifeng, System health management for unmanned aerial vehicle: conception, state-of-art, framework and challenge, in: 2013 ieee 11th international conference on electronic measurement & instruments, volume 2, IEEE, 2013, pp. 859–863.
[6] P. Moosbrugger, K. Y. Rozier, J. Schumann, R2u2: monitoring and diagnosis of security threats for unmanned aerial systems, Formal Methods in System Design 51 (2017) 31–61.
[7] K. Y. Rozier, J. Schumann, C. Ippolito, Intelligent hardware-enabled sensor and software safety and health management for autonomous UAS, Technical Report, 2015.
[8] J. Schumann, I. Roychoudhury, C. Kulkarni, Diagnostic reasoning using prognostic information for unmanned aerial systems, in: Annual Conference of the PHM Society, volume 7, 2015.
[9] S. Zermani, C. Dezan, C. Hireche, R. Euler, J.-P. Diguet, Embedded context aware diagnosis for a uav soc platform, Microprocessors and Microsystems 51 (2017) 185–197.
[10] M. Taylor, J. Boubin, H. Chen, C. Stewart, F. Qin, A study on software bugs in unmanned aircraft systems, in: 2021 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2021, pp. 1439–1448.
[11] D. Wang, S. Li, G. Xiao, Y. Liu, Y. Sui, An exploratory study of autopilot software bugs in unmanned aerial vehicles, in: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2021, pp. 20–31.
[12] Px4, Accessed: October 2023. URL: https://px4.io/.
[13] ArduPilot, Ardupilot, Accessed: October 2023. URL: https://ardupilot.org/.
[14] J. Garcia, Y. Feng, J. Shen, S. Almanee, Y. Xia, Chen, Q. Alfred, A comprehensive study of autonomous vehicle bugs, in: Proceedings of the ACM/IEEE 42nd international conference on software engineering, 2020, pp. 385–396.
[15] F. Zampetti, R. Kapur, M. Di Penta, S. Panichella, An empirical characterization of software bugs in open-source cyber–physical systems, Journal of Systems and Software 192 (2022) 111425.
[16] Federal aviation administration, 2013. Federal Aviation, Regulation. Part-91.