Risk Analysis Report

Author: William Brian Richmond

Institution: Champlain College

Course: SDEV-590: Software Engineering and Project Management Capstone

Instructor: Nathan Braun

Date Due: 09 November 2025

Date Submitted: 09 November 2025

1. Introduction

This Risk Analysis Report builds on the BitGrid OS Capstone, a modular AI-integrated operating system operating at the bit level. It identifies technical, operational, security, scope, and timeline risks and outlines mitigation strategies grounded in PMI's Risk Management Framework and Agile/DevSecOps principles. Continuous tracking through Jira and a living risk register supports iterative monitoring and early detection.

2. Identified Risks and Mitigation Strategies

The BitGrid OS project involves integrating multiple hardware simulation, AI, and software management components within a single operating environment. This complexity introduces several categories of risk that require structured mitigation planning. The following subsections outline five primary risks—technical, security, operational, scope, and timeline—along with their likelihood, impact, and mitigation strategies.

2.1 Technical Risk – Simulation and Integration Complexity

Developing a bit-level computing simulation in Python introduces high integration risk between modules such as the BitIQ AI layer, the BBPU simulation engine, and the security subsystem.

Likelihood: High **Impact:** Critical

Mitigation Strategy: Implement modular prototypes and validate each independently before integration. Apply Agile iteration cycles with incremental builds and use automated unit testing aligned with IEEE 829 standards to identify incompatibilities early.

Tools/Frameworks: Jira (sprint tracking), GitHub (version control), and Lucidchart (architecture validation).

2.2 Security Risk – Vulnerabilities in AI and Communication Layers

Embedding AI-driven optimization into system-level code can expose the system to unauthorized data access and flawed privilege handling.

Likelihood: Medium **Impact:** Critical

Mitigation Strategy: Apply NIST SP 800-53 and Zero-Trust Architecture (ZTA) principles to define access boundaries. Conduct code scanning and peer reviews for every sprint, and simulate penetration testing across communication layers to validate module isolation.

Tools/Frameworks: NIST RMF, STRIDE model, Jira risk register.

2.3 Operational Risk – Resource and Scope Management

As a single-developer project, achieving workload balance and meeting the schedule represents significant operational challenges.

Likelihood: High **Impact:** Moderate

Mitigation Strategy: Use a Kanban board in Jira to visualize work in progress and prevent overload. Establish weekly retrospectives to monitor capacity and progress metrics, including the Cost Performance Index (CPI) and Schedule Performance Index (SPI). Defer non-essential features to a controlled backlog to avoid scope creep.

Tools/Frameworks: Jira Kanban, CPI/SPI tracking sheet, weekly retrospectives.

2.4 Scope Risk – Overextension Beyond Simulation Feasibility

Expanding development beyond the hardware level or into feature overreach could delay the project and exceed Capstone parameters.

Likelihood: Medium **Impact:** Moderate

Mitigation Strategy: Define clear project boundaries in the charter and track all potential scope

changes in a change-control log. Evaluate each proposed addition against Capstone objectives and available resources before approval.

Tools/Frameworks: Risk register, project charter sign-off, Gantt chart (MS Project).

2.5 Timeline Risk – Learning Curve and Tool Integration

Combining multiple design and project management tools (Jira, Lucidchart, Python simulators, and Excel dashboards) adds setup time and integration overhead.

Likelihood: Medium **Impact:** Moderate

Mitigation Strategy: Allocate buffer time within the schedule for configuration and learning. Automate repetitive reporting tasks and use time-boxed sprints to maintain consistent progress.

Tools/Frameworks: MS Project (timeline control), Jira sprint calendar, automated reporting scripts.

3. Risk Monitoring and Conclusion

Risk monitoring will follow an iterative Agile model using Jira dashboards and an Excel risk register with conditional formatting to highlight severity. Weekly retrospectives include a risk-review segment to log new issues, retire closed ones, and assess mitigation effectiveness. Metrics such as burn-down rate and defect-containment index ensure continuous improvement. This disciplined, DevSecOps-aligned approach keeps BitGrid OS within scope and on time.

References

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill.

National Institute of Standards and Technology. (2023). NIST SP 800-53: Security and privacy controls for information systems and organizations. U.S. Department of Commerce.

U.S. Department of Defense. (2022). *DoD Software Modernization Strategy.*

Heusser, M., & Larsen, J. (2023). Software testing and continuous quality.

Champlain College Online. (2025). Week 3: Identifying and managing risks in software projects.