# The Requirement for Software Engineers at the Tactical Edge

## Developing an organic capability

by Maj David McGee

***>Maj McGee is a Communications Officer currently assigned to the Marine Coders Section in the Marine Innovation Unit. He previously deployed as the Officer in Charge of the Deployable Joint Command and Control Detachment 20.2. He served as the Operations Officer for Marine Wing Communications Squadron-48 Det A Fwd, the S-6 Director for Marine Corps Air Station Yuma, and held Platoon and Company Command at 8th Communications Battalion. He works as a Computer Scientist at the Los Alamos National Laboratory.***

In the spring of 2022, I met a group of Marines who shared a common hypothesis that the Marine Corps needs uniformed software engineers at the tactical edge, deployed to support battalion and regimental objectives. I was quickly convinced that developing an organic Marine Corps software engineering capability would be a critical line of effort for success in future conflicts. Our future opponents will be prepared to conduct systems warfare, from offensive cyberspace actions to targeting our systems and supply chains to conducting tactical actions with automated/autonomous systems that make realtime path or targeting decisions, such as drone swarms. Software engineers at the tactical edge can reduce the time for decision-making cycles, improve command and control (C2)/communications capabilities, provide decisive capabilities in an offensive and defensive cyberspace, and enable long-term maintenance and support for systems in denied or restricted communications environments.

Software engineering capability development should become one of the Marine Corps' critical efforts as it executes *Force Design* and thinks deeply about the mechanics of combat in the Pacific. We are shifting our focus toward an adversary that has significant systems development and manufacturing capabilities. We must increase our effectiveness in that environment. Developers at the point of use, conducting actions such as machine-learning model retraining and software development, can provide a tactical advantage. A trained software engineer, capable of adjusting and configuring systems to detect new signatures or new adversarial techniques, can add significant capabilities to a commander.

Software engineers at the tactical edge can improve decision making by changing systems. Software is the natural point for system customization and adaptation, especially when extended physical supply chains limit hardware customization. Software can be adapted to handle new inputs as enemy signatures or battlefield conditions change.[1] These adaptations can drive decision-making improvements. Take an example from the war in Ukraine, where autonomous drones are retrained on videos of Russian troops to enable them to better target soldiers based on their uniforms and behavioral patterns.[2] Adapting and modifying the drone's software improved its target orientation abilities as well as its capability to make decisions. The next war we fight will require systems adaptation throughout the conflict. Just like in Iraq, where V-hulled vehicles became a priority after insurgents started using improvised explosive devices, we will need to make offensive and defensive system adjustments on the fly. Tactical software engineers are crucial to adapting systems to improve decision cycles.

Software engineers can improve C2/communications capabilities. The systems providing C2 rely on software. Engineers who can improve those communications systems can make a significant difference. Such improvements include automated programming interface integrations to share data between maintenance and operational planning systems, integration of sensors into a common operational picture, or building tools for handheld devices. Command and control will be a target for a near-peer adversary, and strategies for enabling C2, such as hiding encrypted messages in the noise of a civilian communications spectrum, will be essential. Software engineers can allow commanders to achieve an advantage by adding to and integrating existing information systems, sensors, and sources.

Software engineers at the tactical edge provide decisive capabilities in an offensive and defensive cyberspace. Just as network administrators limit network access with firewalls and systems administrators restrict access to systems through exposed ports and protocols, a software engineer can add input validation, modify decision algorithms, and update vulnerable code or libraries. A software engineer can conduct offensive scripting, cyberspace reconnaissance, enable vulnerability detection, and conduct exploitation. Decentralizing that capability to the tactical edge allows it to be employed to accomplish the commander's objectives and reduces the reach-back requirement for an organization like MARFORCYBER.[3] Such reach-back and coordination may be impossible in a communications-denied environment, much less the risk of differing priorities at the functional combatant command level. Software engineers at the tactical edge can enable cyber effects in offensive and defensive cyberspace.

Software engineers shorten the software supply chain and can provide long-term maintenance and support capabilities for denied communications and restricted internet communications. How will we defend the network we call the internet in a conflict with an adversary who maintains a closed internal network?[4] The challenge in such a conflict will be sustaining software and systems over an extended period when the pipeline's integrity between the commercial developer and the Marine with the problem is severed. Maintaining systems and applications will require software updates and critical security patches. The level of effectiveness the Marine Corps can maintain over time may be based on the ability to modify applications, systems, and infrastructure to respond to threats. For example, a known compromise of an encryption algorithm might require internal library updates to an application that could be the difference between functioning securely and not functioning at all. Who—on an island in the Pacific—will do these critical updates? Reaching back to the mainland for support will be difficult or impossible. Can we train Marines to do this?

The answer is yes. One such software engineer is Sgt William Crum. While deployed, he built an application based on log messages to analyze a ship's engineering plant malfunction. Additionally, he was able to modify a Python script in theater to change the state of a tactical system. His ability to modify the underlying logic that the system used to function prevented redeploying the associated hardware. This is an example of how a trained software engineer, at the point of use, can solve problems that would otherwise require an extended support chain, maintaining availability without requiring the systems to be shipped to the rear. And it bears repeating that the next conflict will likely occur somewhere where hardware redeployment may not be possible.

***Software engineers shorten the software supply chain and can provide long-term maintenance and support capabilities for denied communications and restricted internet communications.***

If we intend for our systems to function reliably in the future fight, we need to consider deploying software engineers alongside them.

In addition to sustainment and longevity, software engineers can add capabilities through automation. Capt Ryan Helm successfully developed

machine learning models for predicting helicopter landing zones, movement routes, and other tactical problems. Capt Helm's efforts have focused on the ability to accelerate and improve decision making by providing alternatives or potential solutions to a decision maker. Such decisions are made in rapidly changing environments that must incorporate the latest information. A squad leader can implement a virtual terrain model for rapid squad or platoon orientation in significantly less time than it would take to build a terrain model and with significantly greater precision, improving terrain recognition using current imagery. Instead of "take a right at this cardboard box," the squad leader can play the route in realtime and point out that they need to "take a right after the off-white house with the broken window." A platoon commander can accelerate tactical decision making and execution by predicting likely movement paths using elevation, terrain, and vegetation data. Then they can confirm whether his unit will actually be able to use those paths, instead of trying to do so solely based on elevation information on a two-dimensional map. Then they can use the leader's reconnaissance to validate the path, rather than explore several options. These are a couple of the ideas that Capt Helm proposed and implemented. A software engineer who understands the tactical problems they are trying to automate is uniquely poised to implement software solutions.

A third example of a Marine refining software to solve tactical problems is Maj Kevin Hannay, a PhD in Mathematics from Michigan. He refined a K-Means clustering algorithm that Capt Helm used for helicopter landing zone prediction into a convolution algorithm that increased prediction accuracy and reduced calculation times to enable the program to be deployed on a tablet. Maj Hannay made Capt Helm's idea possible to implement on a handheld device accessible to Marines in theater, closing the gap to tactical implementation. Sgt Crum, Capt Helm, Maj Hannay, and others demonstrate what software development at the tactical edge can provide the warfighter. They also illustrate the importance of uniformed software engineers developing solutions to problems they understand and have experienced.

How can the Marine Corps institutionalize or scale this native software engineering capability beyond individual efforts? The Marine Corps possesses several existing pipelines to develop software engineers, from graduating Marines with computer science degrees annually via the Naval Postgraduate School to more applied programs.[5] The recently formed Marine Corps Software Factory pilot program, led by LtCol Charlie Bahk, demonstrates the ability to train enlisted personnel to develop software relatively quickly.[6] This may partly stem from the fact that the typical Marine Corps recruit is far more comfortable writing code than their senior leadership may realize. More than half the high schools in the United States now offer a computer science course.[7] This is a growing skill set that the Marine Corps should leverage in the future via recruitment and incentives. The Marine Corps Software Factory should be formalized and expanded from a pilot program into a critical part of the organizational software development capacity growth.

Two other organizations are currently centers of software development activity in the Marine Corps. The first is the Marine Corps Tactical Systems Support Activity, which recognized the need for software developers, employing them internally to develop multiple tactical applications. The second is the Marine Coders, a community of practice and a section inside the Marine Innovation Unit, which provides a reserve uniformed software engineer capability aligned to support Marine Corps Software Factory projects.[8] This unit can match academic and commercial skill sets to Marine Corps problems. The Marine Innovation Unit could offer a significant strategic capability in times of war, leveraging the reserve component as a strategic source of high-demand, low-density skills and abilities.

Software engineers will drive future strategic and tactical successes against near-peer threats. The Marine Corps currently possesses limited resources but could effectively use existing organizations, such as the Marine Innovation Unit or the Marine Corps Software Factory to field Marines with appropriate skill sets. The Marine Corps can build a cadre of talented engineers who can adapt their skills to differing platforms and applications. Software development at the tactical edge is a critical bid for success in future conflicts.

**Notes**

1. James Marson, "Ukraine Advances Killer Robot Drones With More Automation, Efficiency," *Wall Street Journal*, December 31, 2024, https://www.wsj.com/world/ukraine-advances-killer-robot-drones-with-more-automation-efficiency-0ab132a6.

2. Staff, "Ukraine Trains Army of Robot Drones to Identify Russian Troops by Movement and Uniform," *Forces News*, October 11, 2024, https://www.forcesnews.com/ukraine/ukraine-training-army-robot-drones-identify-russian-troops-movement-and-uniform.

3. U.S. Marine Corps Forces, Cyber Command, "Our Mission," *Marines.mil*, n.d., https://www.marforcyber.marines.mil.

4. Staff, "Internet," *Britannica*, June 5, 2025, https://www.britannica.com/technology/Internet.

5. Staff, "Computer Science," *Naval Postgraduate School*, n.d., https://nps.edu/web/cs.

6. U.S. Marine Corps Software Factory, "Who We Are," *Marines.mil,* n.d., https://www.information.marines.mil/Units/Marine-Corps-Software-Factory.

7. Jeffrey R. Young, "Computer Science Course Offerings in High School Spur More Students to Coding Degrees," *EdSurge*, February 6, 2024, https://www.edsurge.com/news/2024-02-06-computer-science-course-offerings-in-high-school-spur-more-students-to-coding-degrees.

8. Staff, "Marine Coders," *Marine Coders*, n.d., https://marines.dev.