Strategic Release Engineering at General Dynamics IT

By Mark J. Hogan

Senior Architect | Infrastructure Strategist | Technical Mentor

Table of Contents

1. Automating Enterprise Release Workflows with C# and TFS

At General Dynamics IT, release coordination was a high-friction process—manual, repetitive, and vulnerable to error. I led the design and development of Release Manager, a C# WinForms application tightly integrated with Microsoft TFS. It provided a GUI-driven interface for selecting release templates, generating work order tickets, and orchestrating deployment steps across teams.

By embedding logic into the tool itself, we eliminated ambiguity and accelerated release velocity. Engineers could focus on validation and oversight, not procedural minutiae.

2. Architecting Dependency-Aware Ticket Progression

Releases often involved multiple interdependent tickets. To enforce procedural integrity, I engineered a dependency model within Release Manager: no ticket could progress to “Ready” until its predecessors were completed. This logic was monitored in real time, with automated status transitions triggered by upstream completion.

The result was a self-regulating release pipeline—one that respected sequencing, reduced manual oversight, and surfaced blockers early.

3. Streamlining Artifact Validation and Deployment Accuracy

Artifact consistency was critical. I implemented automated routines to gather binaries, configuration files, and documentation from source control and file servers. These routines validated naming conventions, version alignment, and completeness—ensuring that every deployment was backed by a traceable, verified artifact set.

This reduced post-deployment issues and created a reliable audit trail for compliance and rollback.

---

4. Time Sheet Manager: Normalizing Work Hours from TFS APIs

Microsoft TFS tracks total hours per ticket but lacks granularity for daily task breakdowns. I developed Time Sheet Manager, a C# WinForms application that allowed employees to log daily work descriptions across one or more assigned tickets.

Using TFS APIs, I crafted a normalization algorithm that transformed total logged hours into an in-memory table of tasks per day. This enabled granular reporting of hours worked per ticket per day—simplifying time tracking, improving accuracy, and reducing administrative overhead.

---

5. Automating ArcGIS Service Deployments with REST and Python

Geospatial service deployments were traditionally manual and error-prone. I automated the rollout of ArcGIS services using their REST APIs and Python scripting. This allowed for repeatable, scalable deployments with built-in validation and rollback logic.

The automation reduced deployment time, improved consistency, and allowed engineers to focus on higher-value tasks.

---

6. Reflections on Scalable Internal Tooling and Human-Centered Design

My work at GDIT was driven by a belief that internal tools should be as thoughtfully engineered as customer-facing systems. Release Manager and Time Sheet Manager weren't just utilities—they were strategic enablers of clarity, consistency, and human dignity in engineering workflows.

By automating the repetitive and surfacing the invisible, I helped teams focus on what matters: building resilient systems and preserving operational legacy