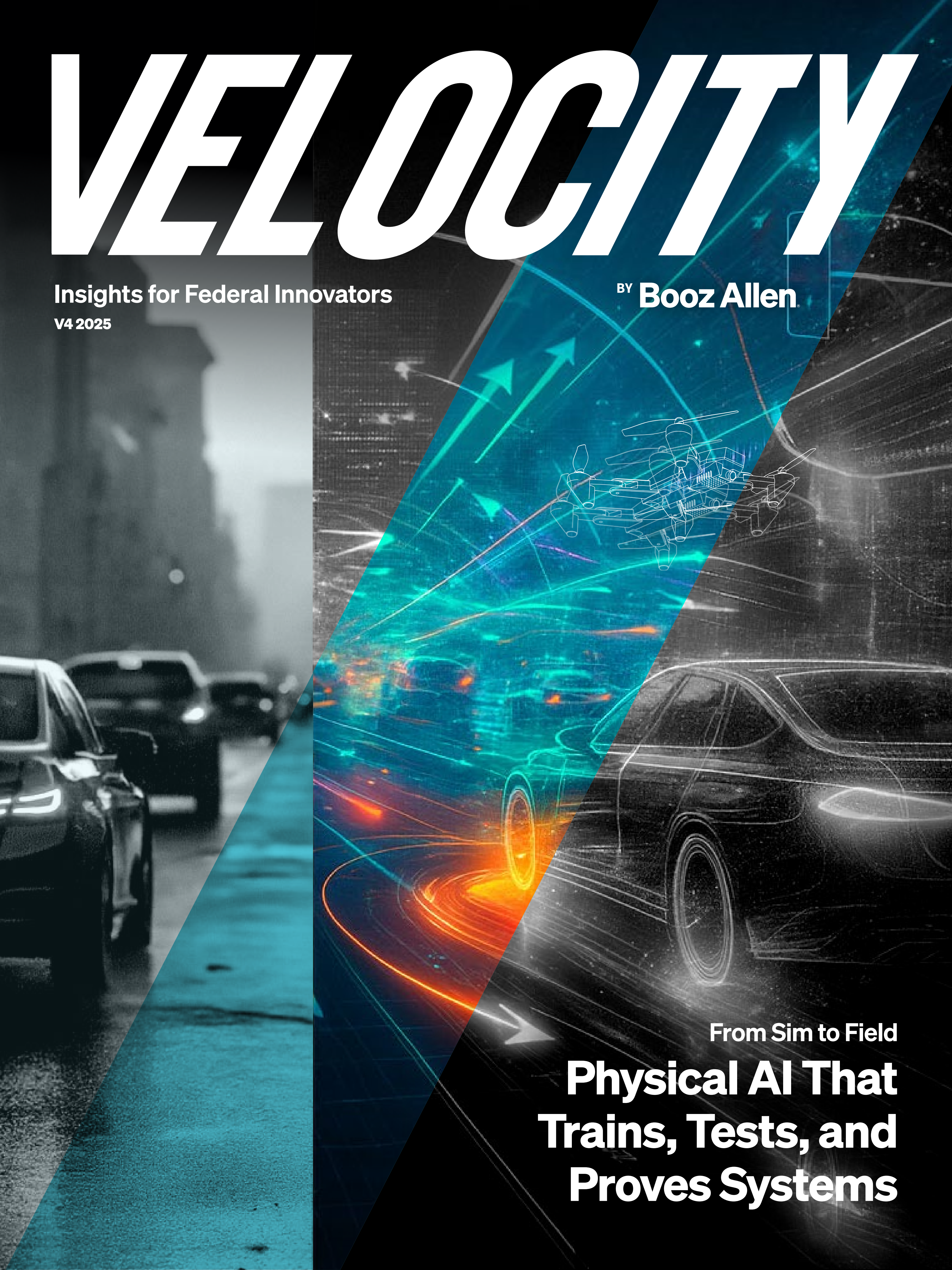
VELOCITY

Insights for Federal Innovators

V4 2025

BY Booz Allen



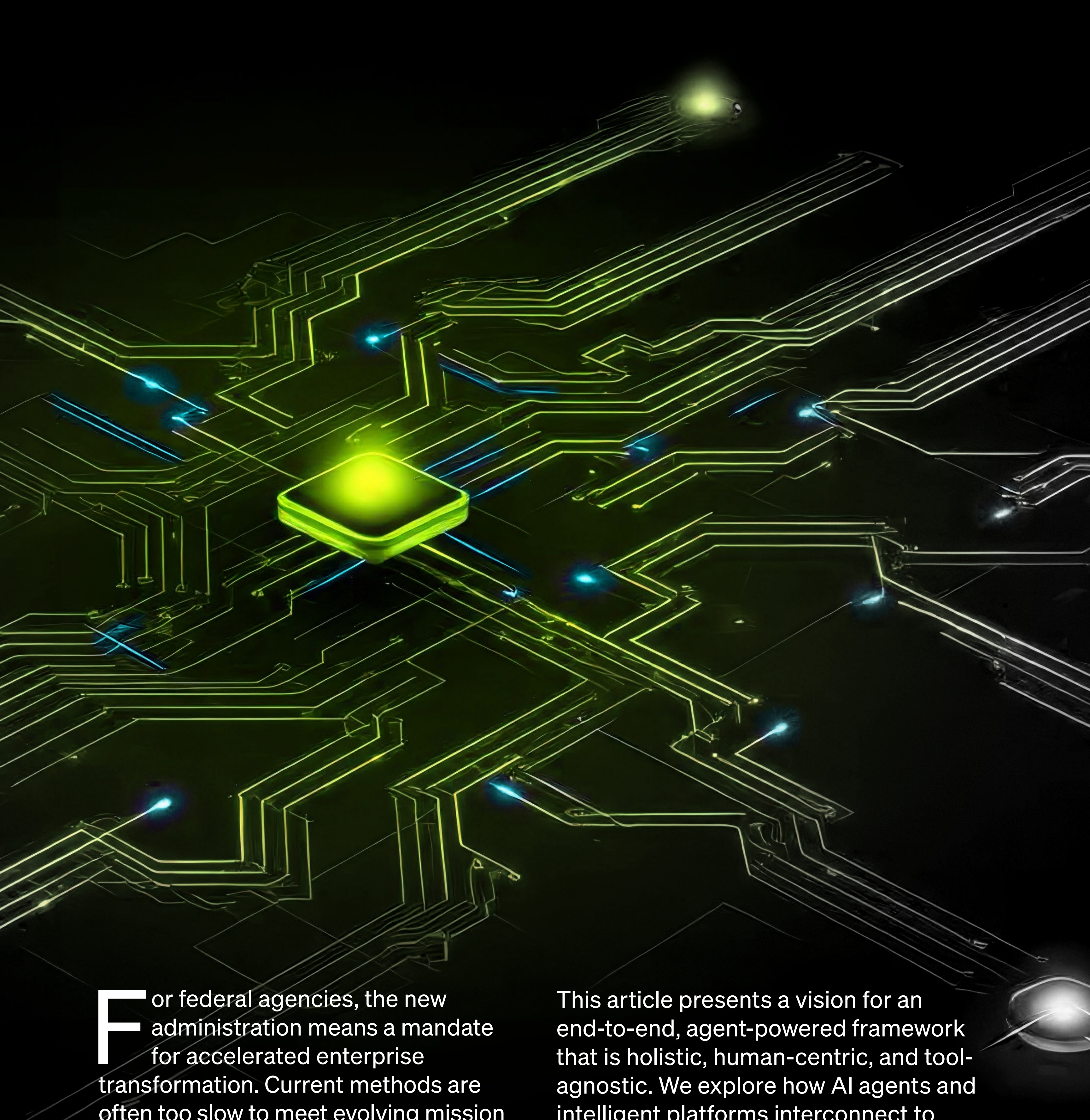
From Sim to Field

Physical AI That Trains, Tests, and Proves Systems

Agentic Development Decoded: Hard-Won Lessons From the Trenches

How federal IT teams can compress the software development lifecycle, accelerate delivery, and break the legacy transformation logjam.

Bassel Haidar, Ryan Stone, Jennifer Sheppard, and Michael Martoccia



For federal agencies, the new administration means a mandate for accelerated enterprise transformation. Current methods are often too slow to meet evolving mission demands, and too resource-intensive for agency budgets. While AI-powered code generation offers a significant tactical boost, the true transformative potential of AI lies in automating nearly every phase of the entire software development lifecycle (SDLC).

This article presents a vision for an end-to-end, agent-powered framework that is holistic, human-centric, and tool-agnostic. We explore how AI agents and intelligent platforms interconnect to automate everything from requirements analysis to operational monitoring, using a real-world case study to illustrate the path from legacy systems to future-ready, user-centered applications that can finally move at the speed of mission.

The Weight of Legacy and the False Dawn of AI Coding

In an era of accelerating peer competition and rising citizen expectations, fast delivery of secure and reliable software isn't just essential to the mission; it is the mission. For federal leaders, the core challenge is clear: The current approach to the SDLC is too slow, too brittle, and too burdened by manual compliance to keep pace. The distance between a new requirement and a deployed capability is measured in months or years, a delay we can no longer afford.

A new generation of federal leaders with backgrounds in technology or business are determined to reshape the organizations they lead. Agencies and departments are expected to do more with less and are looking to use technology to square that circle.

When it comes to modernization, the message is clear: No more band aids. There's a preference for radical, transformative solutions and greenfield builds.

Against this backdrop, first generative AI and now agentic AI are transforming the development of software, advancing over the past year at a pace that has far exceeded our most optimistic expectations (visit *The Age of Agentic AI* in the previous issue of *Velocity* for a primer on these technologies). For example, the extraordinary speed of improvement of the large language models (LLMs) powering the Claude Code, Cursor, and Github CoPilot tools means that any performance numbers we could include in this article would be out of date by the time it was printed.

It's hard to overstate the significance of these changes. AI has moved from a clever coding accelerant to a broader change in how software gets made. The biggest gains don't come from drafting snippets in an integrated development environment (IDE); they come from shortening the distance between intent and impact. The goal is no longer just improving individual productivity; it's about building an AI-powered software factory.

We believe there's synergy between these vast ongoing changes: The emerging AI-powered software development ecosystem, properly managed, will make the transformation of federal agencies—or any other coding project—possible at a speed and scale unimaginable just a year ago.

Indeed, it is already starting to do so.

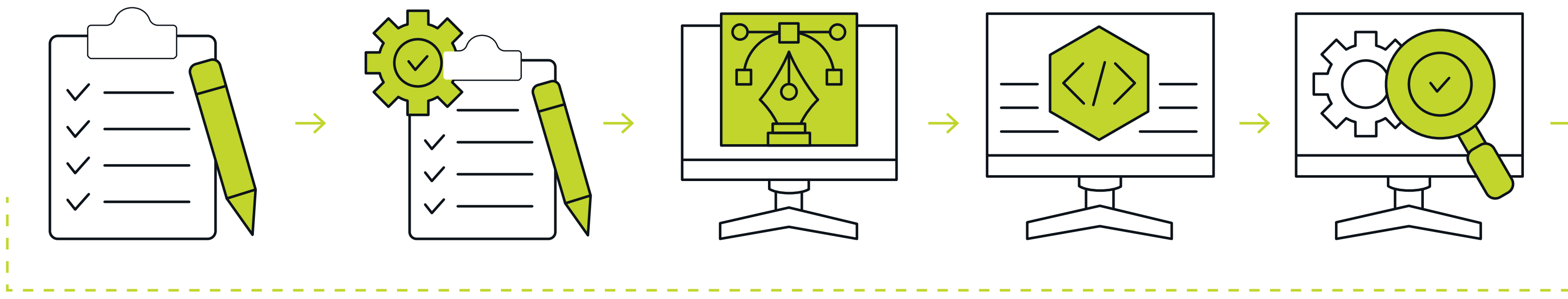
As we write, in September 2025, AI tools are highly effective for automating certain discrete tasks in the software development process. Over the next two years, this paradigm shift will only accelerate and grow.

What We've Learned: From Early Pilots to Durable Platforms

Early pilots and projects across federal agencies and mission partners tell a consistent story. The center of gravity has shifted from "AI coding" to "AI-enhanced delivery." Agencies are experimenting with assistants in IDEs, but the real leverage appears when AI improves upstream clarity and downstream assurance.

The immediate wins are clear and compelling:

- **Code understanding and navigation:** Developers use AI to summarize unfamiliar modules, trace dependencies, and generate scaffolding and documentation that would normally take hours.
- **Test generation and maintenance:** AI accelerates unit, integration, and regression tests, including self-healing scripts tied to selectors and application programming interfaces (APIs).
- **Knowledge capture:** Teams use AI to generate living runbooks, design notes, and architectural decision records from sources, tickets, and wikis.
- **Operational toil:** Incident analysis, log summarization, and safe rollout plans benefit from AI that is connected to continuous integration (CI)/continuous delivery (CD) pipelines and observability platforms.



However, real gaps remain, meaning results were too often uneven:

- **Whole-app generation:** AI can draft simple apps and boilerplate, but complexity, security rules, and legacy interfaces limit end-to-end automation.
- **Context gaps:** Assistants without access to the right repositories, APIs, and policies deliver generic answers that look polished yet require rework.
- **Brittle integrations:** Promising pilot agents become fragile in production when they rely on ad hoc scripts or cannot survive dependency changes.
- **Change management:** Productivity gains fade when teams do not update code review norms, branching strategies, and release gates for AI-touched work.

These early successes share the same profile. They are high-volume, high-friction, low-risk activities where the definition of "good" is clear and verification can be automated. This is exactly where federal programs should start:

1. **Guardrails:** Policy, privacy, and provenance built into the platform.
2. **Grounding:** Access to the right code, data, and documentation—at the right time.
3. **Granularity:** Agents scoped to well-defined tasks with clear acceptance criteria.
4. **Good measures:** Delivery and quality metrics that show where the gain actually is.

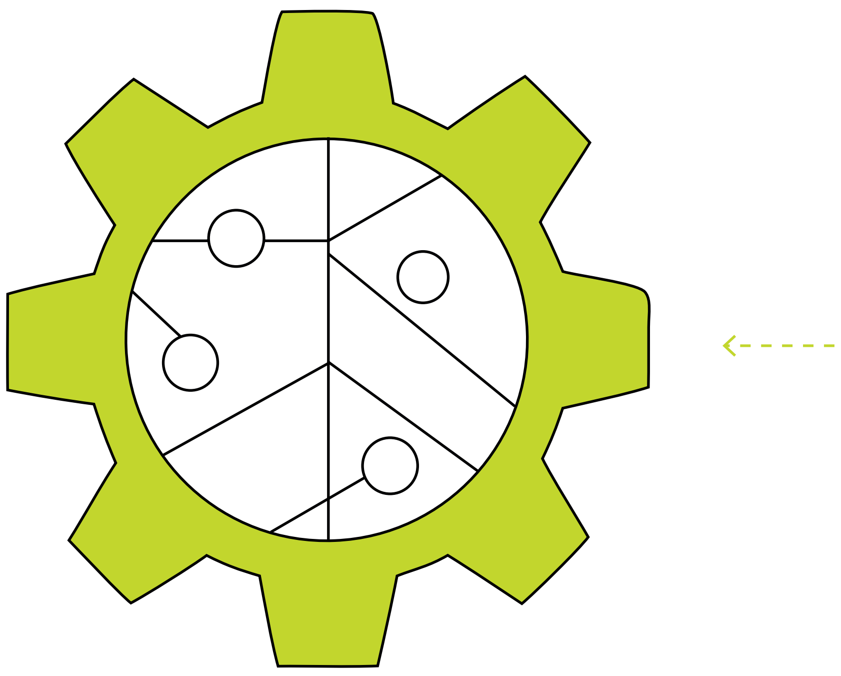
Select external signals confirm the direction. Adoption is rising, agentic patterns are emerging, and the SDLC is compressing in places. At the same time, organizations report uneven results when pilots skip governance, skip measurement, or overreach for full automation.

How AI Compresses the Full SDLC

Let's start with a global perspective. Gartner® predicts that:

"By 2028, 90 percent of enterprise software engineers will use AI code assistants, up from less than 14 percent in early 2024. The role of developers will shift from implementation to orchestration, focusing on problem solving and system design, and ensuring AI tools deliver high-quality outcomes. To succeed, teams must balance automation with human oversight, considering business criticality, risk and workflow complexity."

We believe this mirrors the increasing use of AI-powered approaches to compress the current SDLC that we are witnessing. A starting point is the use of generative AI to automate rote coding tasks. However, agentic AI is increasingly being used to execute more complex assignments, building upon autonomous, outcome-driven AI agents that continuously learn and adapt to achieve goals with minimal human intervention.



The situation is fast-changing, fluid but clear: Traditional manual development is rapidly transforming into human-guided, AI-powered execution. Digging deeper, we find this transformation evolving as follows.

- **Upstream: Intent to Architecture**
  AI agents turn unstructured inputs into a living backlog, cluster related needs, flag conflicts, and link requirements to acceptance tests. Design agents propose patterns, enumerate trade-offs, and stress-test options against nonfunctional constraints such as privacy, safety, and performance. Human architects still decide, but they decide faster and with better evidence.

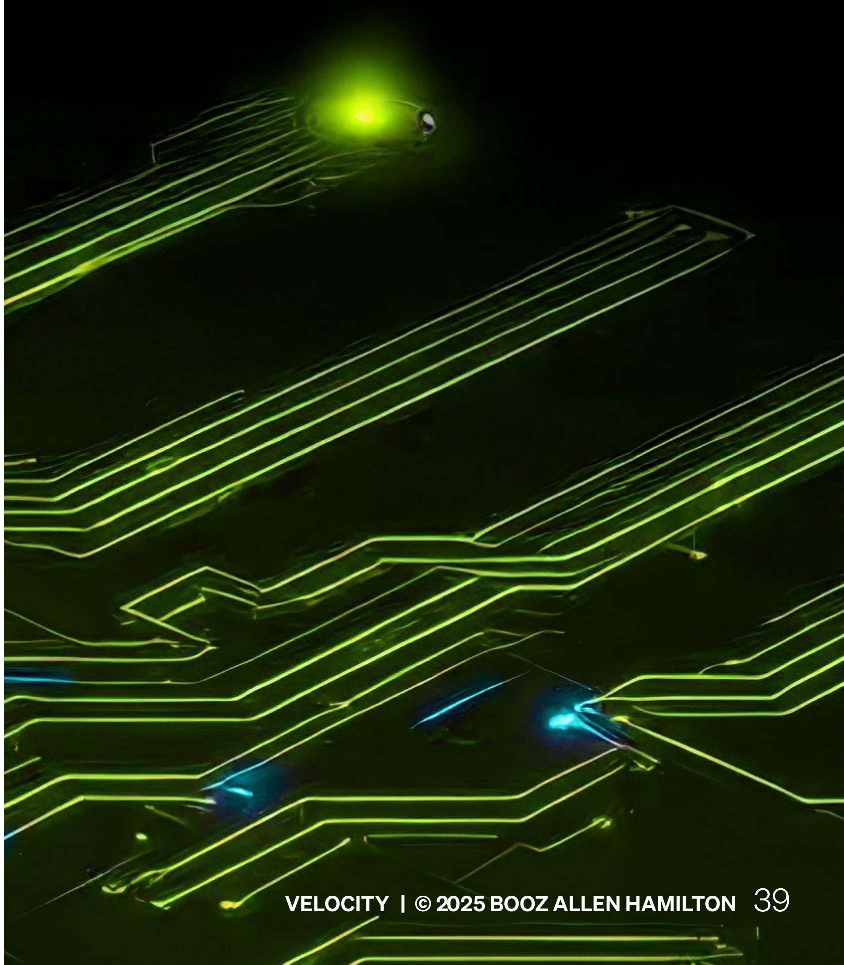
- **Midstream: Build to Assurance**
  Leading teams use a curated workbench with approved APIs, software development kits (SDKs), and templates that encode security and compliance by default (see more in the ***Velocity* article** "How We Built a Better Innovation Factory"). Shift-left becomes real as agents expand test coverage, generate edge cases, and keep tests current as code changes. Policies and security rules are expressed as code and enforced as automated pre-merge checks.

- **Downstream: Operations to Learning**
  In production, agents assist site reliability engineers with anomaly detection, impact analysis, and rollback planning. Connected to playbooks, logs, and topology maps, they recommend the next best action during incidents and feed lessons back into requirements and tests. The aim is safer changes, fewer pages, and faster recovery.

A key differentiator is orchestration across the lifecycle. Software teams add outsized value by mastering these tools, knowing their limits, and governing them well.

Cyber Teams Embrace Agentic Engineering to Defeat the Threat

Agentic AI is emerging as a gamechanger across many facets of software engineering. For example, the Booz Allen **Vellox Reverser™** product is an AI first, cloud native malware reverse engineering product designed to radically speed up threat response. It leverages agentic AI and machine learning to automate an exhaustive analysis of complex malware, delivering deep insights and actionable reports with comprehensive countermeasures in just minutes. AI agents orchestrate dynamic submissions across various tools and sandboxes to quickly assess and adapt defense strategies based on performance and behavior in real time.



The Agentic Developer: A New Blueprint for IT Transformation

It is time to rethink the end to end SDLC. Over the past 20 years, we've evolved from waterfall to agile to DevOps to today's loosely coupled world of cloud based containers, web services, and APIs. AI is rendering this world obsolete faster than we probably realize.

We can no longer draw lines around existing, often human based functions. Yes, there are compliance or other reasons to wall some functions off. However, these barriers are under constant attack from technology's relentless advances.

Instead, we should be guided by principles: Foundational pillars that reflect our values and shape our strategy:

Holistic: We look at the entire software development lifecycle end to end. And we have a master plan to orchestrate every stage and persona of the process. If you don't have all the puzzle pieces, you can't see the big picture.

Human-Centric Design: The software end user is at the center of everything we do, because the users are the ones doing the mission, and the mission is why we are here. But the human centric approach also emphasizes automation and reusability   to free developers from the drudgery of repetitive, non creative work.

Tool-Agnostic: Avoiding vendor lock in is essential. We must be able to utilize tools from any trusted supplier, because the competition to be "best of breed" is only going to get more intense, and, in an emerging technology field, that contest can shift swiftly in unexpected ways.

When we look at the transformative impact these AI tools are going to have, it's important to break down the SDLC and examine each stage. Different parts of the process will be impacted differently:

Phase 1: Make Knowledge Computable

Adapting to the AI powered SDLC starts with making knowledge computable. Consolidate requirements, constraints, and decisions in a single, queryable knowledge base that cites its sources. A requirements assistant sits on top of this record and answers questions, highlights contradictions, and proposes acceptance tests that trace back to stated needs. When teams can interrogate the truth, they find gaps earlier, align faster, and prevent rework before code exists.

Phase 2: Institutionalize Design Intelligence

Design intelligence comes next. Institutionalize agents that compare architectural options against concrete targets for cost, performance, privacy, and scalability. Treat nonfunctional requirements as living guardrails, not slideware. Fitness functions continuously test architectures as they evolve, while an architectural decision record captures the why behind each choice. Human architects still decide; they do so with stronger evidence and less cycle time.

Phase 3: Build the Intelligent Workbench

Developers need a smart workbench, not a bag of ad hoc tools. Stand up a browser based portal that curates approved APIs, SDKs, templates, and code libraries with clear usage guidance. Include examples, ratings, and guardrails so teams can move fast without creating risk. Instrument the portal, learn what gets used, retire unsafe patterns, and expand what actually accelerates delivery.

Phase 4: Make Assurance Continuous

Assurance becomes continuous when testing shifts from a late gate to a built in practice. Use AI agents to generate unit, integration, property based, and adversarial tests, and to keep those tests current as code changes. Express policies and security rules as code, then enforce them before merge and before deploy. Standardize canaries, automatic rollbacks, and post incident learning loops so production teaches development in near real time.

Phase 5: Manage the Agent Ecosystem

Manage the agent ecosystem with the same rigor you bring to software supply chains. An agentic AI studio curates and secures domain trained agents, sets permissions and data scopes, and defines where a human must approve actions. An agentic AI mesh orchestrates handoffs across requirements, planning, design, coding, testing, and operations. Start with supervised autonomy, expand privileges as evidence of reliability grows, and instrument everything. Maintain audit trails, model bills of materials, bias and performance monitors, and clear incident playbooks. The result is a governed system of work where AI accelerates and humans remain accountable.

Case Study:
Reimagining Modernization

Imagine a federal grant program in the near future. The core client server application is more than two decades old, making the system more brittle and costly to maintain. What's more, the associated business processes and data are often locked in place as well. This means that the agency is reliant on manual processes to report performance and ROI. Requirements for updates are communicated via email. The grant approval process takes three or four months, and data from the app shows that many grant applications are abandoned before being submitted, but with no granularity as to why.

Engaged to modernize the process and the app, an AI-assisted Booz Allen development team deploys. Within days, they can generate artifacts: Requirements, business logic analyses, software pathways. The legacy codebase can either be analyzed and translated into business logic by our specialized code-to-logic engine to prepare it for refactoring or transposed directly into modern code languages by a specially trained code translator AI. Within a week, the team has a detailed roadmap, laying out these options and others for the modernization process.

Within a few months, the modernization is complete. The new system is live. The new, more user-friendly interface has a much lower level of abandoned applications. The app uses AI to flag high-risk applications for additional review, reducing fraud. But despite these additional layers of review, grant approval time is down to 30 days or fewer. An operational agent from our prebuilt agent catalogue now provides real-time, public-facing dashboards on fund distribution, fulfilling transparency and congressional reporting requirements automatically, which frees up the program's staff for other duties.

The Human-Centric Imperative: The New Federal Technologist

It might seem counter-intuitive to say so, but the human developer is front and center of this coming agentic AI transformation. Because of the important limitations of AI coding assistants and these other tools that we've discussed, they still need humans to manage them, to check their work, and to add vital context to their output. Developers need to transition from directly employing their coding skills to leveraging them to assign and assess the work of AI agents. They need to move from being coders to mission engineers, orchestrating and managing the work of trained and specialized AI personas.

Because of the centrality of human software developers, and the way their skillsets need to evolve, workforce issues are key for federal agencies seeking to ride this transformational wave. Currently, adoption of these new tools is spotty, according to Booz Allen internal research. There's uncertainty about their value, but also resistance stemming from broader cultural narratives. Some of that is due to fears about job losses, and some to concern about the costs, benefits, and risks of AI—its growing use of power and water for instance, or the perceived dangers of empowering non-human intelligence.

Whatever the source, management has to confront these doubts and fears head on. These tools won't replace developers; they'll empower them, freeing them from repetitive and uncreative drudge work. The expertise of senior developers will be more important than ever. Only those with deep mission knowledge and advanced coding skills will be able to ask the right questions, and successfully select and direct an ecosystem of different AI tools to solve complex problems. Highlighting the successes of early-adopter teams, and leading by example—with management publicly using the tools in high-profile projects—are two ways of getting this message across.

Even as AI agents become more capable and the agentic AI mesh enables them to interact autonomously with each other, that human expertise will still be essential. The best developers and engineers won't write (or won't need to write) the best code any more, but they will train the best AI agents. In the future, the intellectual property represented by those exquisite prompts and curated training data sets will be one of the enterprise's major assets, just as its codebase is now.

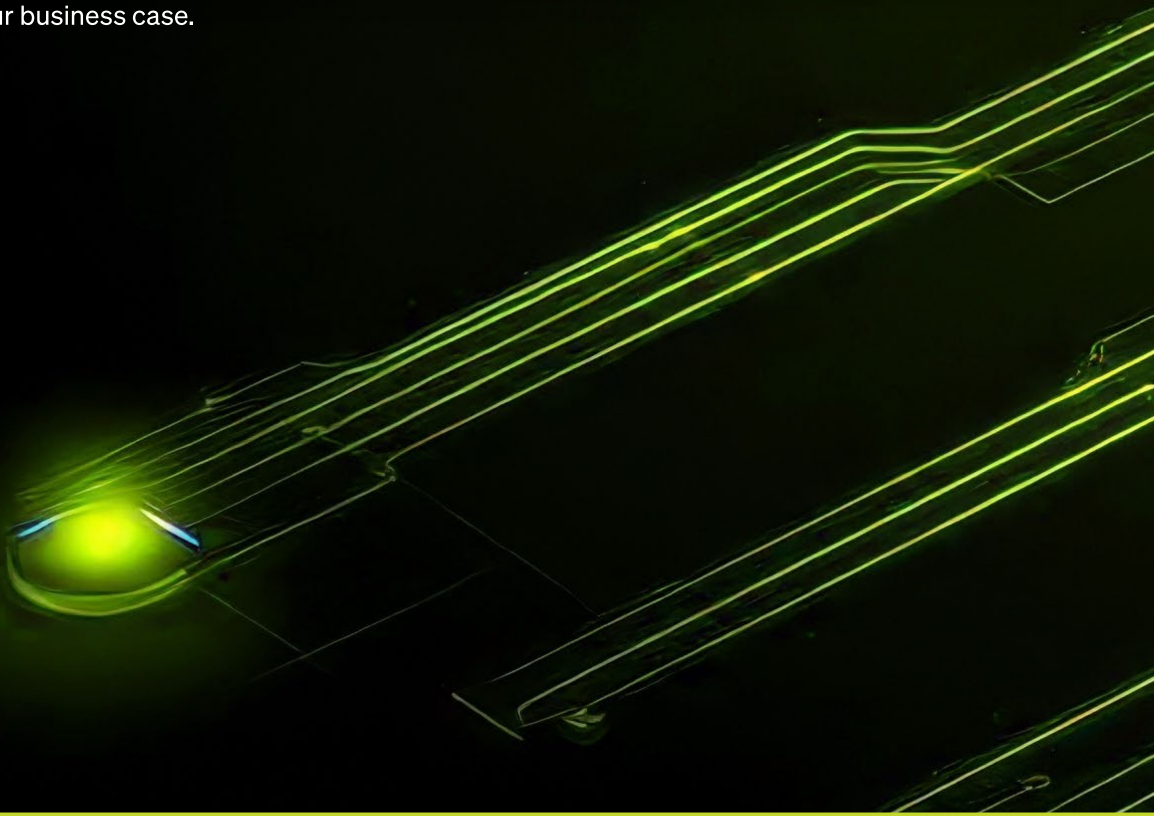
Above all, the message for the workforce should be about how much better these tools make us at doing our job: Enabling the mission. It is about speed, but it's not all about speed. It's all about being able to deliver better, more performant, more reliable software—and do it faster.

Assistive Today, Integrated Tomorrow: Your Roadmap to AI-Powered Software Development

As we write, in September 2025, AI tools are highly effective for automating certain discrete tasks in the software development process. Over the next two years, this paradigm shift will only accelerate and grow. In the commercial sector there will be vendors promising (although not always delivering) end to end automation "out of the box." In the federal context, supervised AI agents will be executing entire chains of activity: Upon receiving a requirement, they will begin drafting user stories, suggesting architectures, generating code, and designing tests, and eventually deploy to a staging environment with minimal human input beyond approval gates. The agent mesh enables software developers to empower "AI teams" chains of agents each executing on the tasks for which they're specially trained and passing off to the next.

Here are specific steps that federal IT leaders can take to prepare their organizations for this brave new world:

1. **Establish Your Baseline:** You cannot justify an investment without proving an ROI. And for that, you need data. Begin by deploying a software engineering intelligence platform to measure your current processes, identify bottlenecks and blockages, and start to build your business case.
2. **Launch a Lighthouse Project:** Select a high visibility, low risk project. Maybe a legacy modernization, maybe greenfield: Whatever is available and fits the required risk/visibility profile. The point is not the project; it's the public testbed for new AI tools and the agentic developer concept. A successful lighthouse project demonstrates their utility and builds organizational buy-in.
3. **Set Realistic Expectations:** The initial ROI is efficiency. The ultimate, strategic ROI is mission agility. The goal is not just to deliver faster, not even to deliver faster and better. Rather, the goal is to build an organization that can create, adapt, and deploy new digital capabilities at the speed of need, not at the pace of a multi year procurement cycle.
4. **Create an AI Center of Excellence (CoE):** This group will be responsible for governance: Interpreting policy and setting standards. They will evaluate and vet available tools, curating a digital workbench of AI tools to ensure developers have a consistent and secure toolset.
5. **Invest in Your People:** Begin training programs now to upskill your workforce, focusing on AI literacy, data analysis, data science, and the new software development and engineering skills required to manage and orchestrate AI systems.



SPEED READ

AI agents can automate nearly every phase of the software development lifecycle (SDLC), enabling federal IT teams to accelerate delivery and modernize legacy systems at unprecedented speed.

A holistic, human-centric, and tool-agnostic approach is crucial for leveraging AI to transform federal software development, ensuring that AI empowers developers rather than replacing them.

Federal agencies can achieve faster, more reliable, and mission-centric software delivery by integrating AI-powered tools, fostering a collaborative innovation environment, and upskilling their workforce.

<UNFLINCHING_COURAGE*CHARACTER*AGILITY*
RESULTS*CODER*FEROCIOUS_INTEGRITY*CHAN
GE*SECURITY*DEFENSE_TECH*AGENTIC_AI*ENGIN
EERING*LEADERSHIP*PASSIONATE_SERVICE*CODE
*SOFTWARE*QUANTUM*MILITARY*SAFEGUARD
*PARTNERSHIP*EFFICIENCY*SCALABILITY*REVOL
UTION*DIGITAL_TWIN*COLLECTIVE_INGENUITY*
OUTCOMES*PATRIOTISM*ADVANTAGE*PIONEER*
DEVELOPER*INNOVATION*VR*PRODUCTS*SOLUTIO
NS*DATA*EVOLUTION*BUILDER*COMMITMENT
*TRANSFORMATION*FUTURE*FORWARD*BREAK
THROUGH*CO-CREATION*CHAMPION'S_HEART*MI
SSIONS*SCALE*CLOUD*CYBER*VELOCITY*COLL
ABORATION*NATIONAL_SECURITY*AUTONOMY/>

0011110001101001011101000111001101011111011010010110111001011111
0110111101110101011100100101111011000110110111101100100011001 01
IN THIS COUNTRY, WE DON'T FOLLOW. WE LEAD. WE DON'T WAIT.
WE MOVE FIRST. WE DON'T WATCH HISTORY HAPPEN. WE MAKE IT.
00111110_0011110001101001011101000111001101011111011010010110111 0
0101111101101111011101010111001001011111011000110110111101100100 01
WHILE OTHERS TALK ABOUT AI...WE PUT IT UNDER THE OCEAN.
ON THE FRONTLINES. AND UP IN SPACE. THAT'S RIGHT—SPACE.
1001010011111 0_00111100011010010111010001110011010111110110100 10
110111001011111011011110111010101110010010111110110001101101110110
HOW DO WE SOLVE OUR NATION'S BIGGEST CHALLENGES? SPEED.
PRECISION. HONOR. NO MANUAL. NO ROAD MAP. NO PROBLEM.
0100011001010011111 0_001111000110100101110100011100110101111101 1
0100101101110010111110110111101110101011100100101111101100011011 01

DEVELOPING 5G VR MILITARY TRAINING. SECURING BORDERS WITH ARTIFICIAL INTELLIGENCE AND TRAILBLAZING TECH.
BECAUSE REAL LEADERSHIP? IT'S ABOUT BUILDING WHAT NOBODY ELSE CAN AND CODING SO WE CAN'T LOSE.
1110110010001100101001111 10_00111100011010010111010001110011010111110110100101101110010111110110111101110101 01110
010010111110110001101101111011001000110010100111110_00111100011010010111010001110011010111110110100101101110 0101
ENGINEERS. CODERS. SPECIAL OPS. WINNING EVERY TIME. AND MAKING AMERICA STRONGER. SAFER. FASTER.
IT'S WHO WE ARE. IT'S WHAT WE STAND FOR. IT'S WHAT WE'RE MADE OF. BOOZ ALLEN <ITS_IN_OUR_CODE>
111101101111011101010111001001011111011000110110111101100100011001010011111 0_111101101111011101010111001001011111011
0001101101111011001000110010100111110_1111011011110111010101110010010111110110001101101111011001000110010100111 10
<ITS_IN_OUR_CODE> <ITS_IN_OUR_CODE> <ITS_IN_OUR_CODE> <ITS_IN_OUR_CODE> <ITS_IN_OUR_CODE>
<ITS_IN_OUR_CODE> <ITS_IN_OUR_CODE> <ITS_IN_OUR_CODE> <ITS_IN_OUR_CODE> <ITS_IN_OUR_CODE>
_001111000110100101110100011100110101111101101001011011100101111101101111011101010111001001011111011000110110111 0
11001000110010100111110_0100001001101111011011110111101000100000001000000101101100011011000110010101101110

Booz Allen®

<its_in_our_code>