NGA THE NGA SOFTWARE WAY

EXECUTIVE SUMMARY

NGA needs to deliver software that meets our users' needs and expectations.

To measure how we meet this intent, there are three key metrics NGA will track for each software product:

AVAILABILITY

LEAD TIME

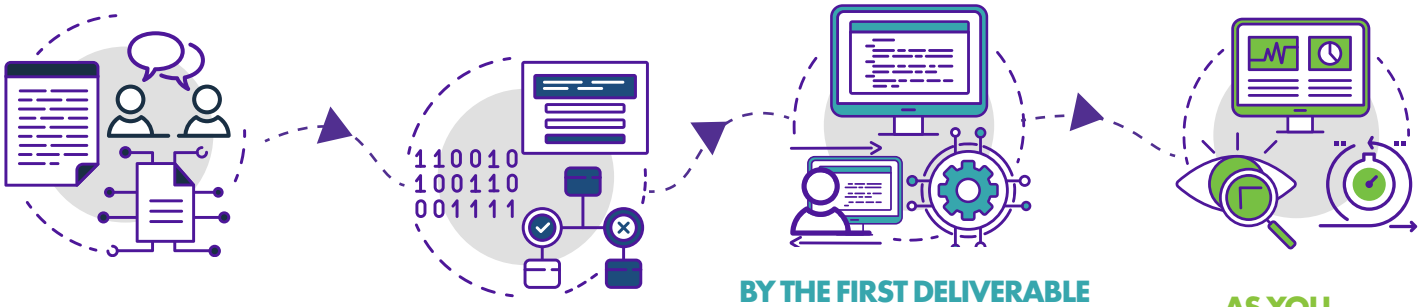
DEPLOYMENT FREQUENCY

Additionally, each software product will be expected to have its own key performance indicators and/or metrics to track how well the product is working for its users.

To be able to consistently track these metrics and realize the above intent, this document provides a set of elements that NGA and contractor teams must meet while building software products.

These elements, collectively referred to as "The NGA Software Way," ensure each software product has the people, processes, and tools in place to safely and effectively operate at NGA.

THE NGA SOFTWARE WAY ELEMENTS



BEFORE WRITING CODE

1. Identify and Empower a Product Owner
2. Deeply Understand User Needs
3. Prioritize Rapidly Delivering Value to Users

WHEN BEGINNING AND THROUGHOUT DEVELOPMENT

4. Use Version Control
5. Track Work Consistently
6. Automate Testing

BY THE FIRST DELIVERABLE THAT PROVIDES VALUE TO THE USER

7. Provide Support and Feedback Mechanisms
8. Document and Share APIs
9. Automate Deployments
10. Automate Monitoring

AS YOU ITERATE

11. Automate Alerting and Incident Response
12. Use Data to Drive Decisions
13. Iterate and Improve Continually

NGA CTO INTENT

Alex Loehr
NGA Chief Technology Officer

NGA needs to deliver software that meets our users' needs and expectations. To do this, the NGA Technology Strategy describes changes in approach, tools, and behavior that will result in faster iterations, more flexible products, and better-satisfied customers.

Our intent is to get to a place where NGA software products are being built with users (not for them), have key performance indicators that are automatically measured (and decisions are based on), and can be updated in production daily (or at the speed that mission demands, whichever is faster). We believe in small, empowered, cross-functional teams that have a deep understanding of and empathy for their users. The teams must work directly with those users on a regular basis to iterate rapidly and ensure the products are meeting their needs.

To understand how teams are doing at meeting this intent, NGA has developed consistent metrics to track across products.

SOFTWARE METRICS

The three initial metrics that all programs will track for each software product delivered to users are availability, deployment frequency, and lead time. It is important to note that NGA is not unique in using these metrics. These are three of the primary metrics DevOps Research and Assessment (DORA) has studied over the past six years. These metrics use data from over 31,000 professionals worldwide, and rigorous statistical methods to study the most effective ways to develop and deploy technology. More information about the DORA metrics and study can be found here.

These metrics provide visibility into how NGA balances speed (deployment frequency and lead time) and stability (availability) during product development and operations. These two efforts, speed and stability, used to be believed to be negatively correlated – increasing one would decrease the other. The DORA research has conclusively shown that speed and stability are outcomes that enable each other and that high-performing teams do better in both of these areas. This is what NGA is aiming for and moving toward.

NGA software products must automatically track these three metrics and share the results in transparent dashboards.

AVAILABILITY

Availability is about ensuring a product or service is accessible to users when they need it. At a macro level, availability represents technology teams and organizations meeting user expectations about the software they are deploying and operating. Any time a user is not able to access a software product, it is unavailable. Availability may be impacted by actions a product team itself takes or it may be impacted through issues with upstream or downstream dependencies. Whether planned or unplanned, and no matter where the issue originated, the impact of a product being unavailable when users need it is the same.

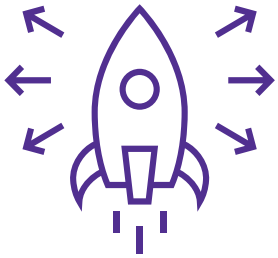
Higher availability is better, as it means the product is accessible when users need and expect it. Tracking availability automatically enables teams to know when something goes wrong, giving the opportunity to investigate issues more rapidly and successfully. Teams may have different specific availability requirements depending on mission and user needs. Teams will be expected to track both internal availability and actual availability for users, reporting on both of those and working with partner teams and organizations to decrease and close the gaps between them.

LEAD TIME

Lead time is the time it takes from when a customer makes a request to when a feature meeting that request is built and available for mission use. In measuring lead time, there are two parts: (1) the "design and engineering lead time" it takes to validate the need and design a feature and (2) the "development and deployment lead time" it takes to build and deliver the feature to users. Together, these are referred to simply as "lead time."

The design and engineering lead time begins when a requirement is received and ends when that requirement is well enough defined for a development team to implement it. The development and deployment lead time begins when the feature is ready to be developed, even if the development team isn't actively working on it yet, and ends when the feature is in production and able to be used by users during an actual mission (not in a test environment). The development and deployment lead time can be further broken down into the following stages: (1) time from when a feature is ready to be developed to when a team begins working on it; (2) time from when a team begins working on it to when it is functioning properly in a development environment; and (3) time from when a feature is functioning properly in a development environment to when it is available in production for mission use. Measuring these various stages enables teams to understand where bottlenecks are in overall lead time and prioritize where to focus on optimizing.

Shorter lead times are better because they enable faster feedback on what is built and allows teams to course correct more rapidly. Additionally, faster lead times allow faster fixes with higher confidence when there are issues, which directly impacts availability as well.



DEPLOYMENT FREQUENCY

Deployment frequency is how often a product is deployed to production. A product team will do many additional deployments to testing, development, staging and other non-production environments. The deployment frequency metric is specifically related to deployments to production, where users actually use the product to complete the mission or function.

Deployment frequency is a proxy for batch size. Reducing batch size has consistently been shown to accelerate feedback, improve efficiency, reduce risk and overhead, increase team motivation and urgency, and reduce costs. (For more details, see Accelerate, page 16, by Nicole Forsgren, Jez Humble, and Gene Kim).

PRODUCT-SPECIFIC METRICS

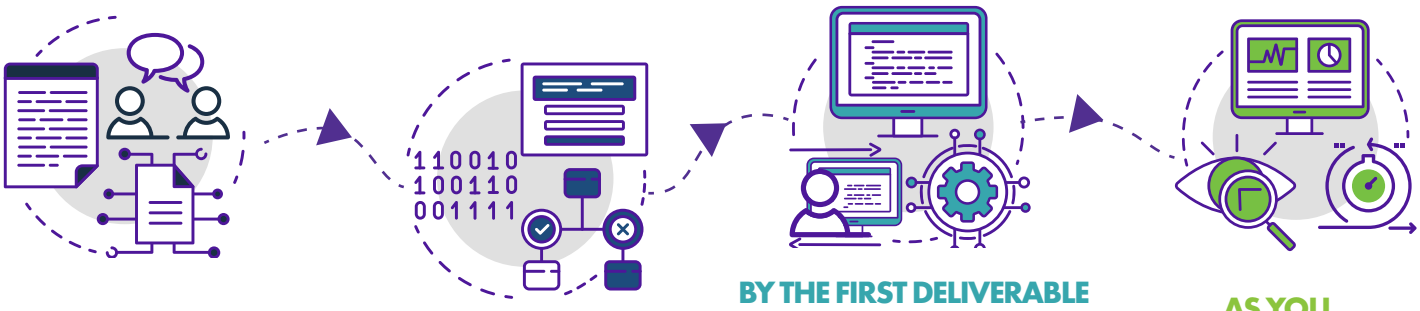
In addition to the above three metrics that will be consistently tracked across all new software programs, each software product will be expected to have its own key performance indicators and/or metrics to track how well the product is working for its users. These product-specific metrics will not be consistent across NGA, and cannot be defined at the organizational level. However, each product will include: (1) performance metrics that describe if the product is doing what it's supposed to; (2) business metrics that tell us how well the product is meeting organization and mission goals; and (3) exploratory metrics to understand how people are using the product and to prioritize feature development. It will be the responsibility of the product manager or product owner to define and communicate these product-specific metrics.

ABOUT THIS DOCUMENT

To be able to consistently track these metrics and realize the above intent, this document provides a set of elements that NGA and contractor teams must meet while building software products. These elements, collectively referred to as **"The NGA Software Way,"** ensure each software product has the people, processes, and tools in place to safely and effectively operate at NGA. They apply to any team building a new product or modifying one already in service, whether making small changes, adding large new features, or creating entirely new products. This document will be updated and republished as both NGA and industry best practices evolve.

"Product" is an overloaded term at NGA. In this document, a "product" is software, rather than a finished intelligence product, and refers to an application or service that is used in accomplishing NGA's mission. Products can be directly user facing or can be backend services that end-users depend on indirectly, but other developers directly depend on to build their products. The NGA Software Way applies to both of these cases, as even for backend products, someone needs to use and depend on that product to deliver on NGA's mission.

THE NGA SOFTWARE WAY ELEMENTS



BEFORE WRITING CODE

1. Identify and Empower a Product Owner
2. Deeply Understand User Needs
3. Prioritize Rapidly Delivering Value to Users

WHEN BEGINNING AND THROUGHOUT DEVELOPMENT

4. Use Version Control
5. Track Work Consistently
6. Automate Testing

BY THE FIRST DELIVERABLE THAT PROVIDES VALUE TO THE USER

7. Provide Support and Feedback Mechanisms
8. Document and Share APIs
9. Automate Deployments
10. Automate Monitoring

AS YOU ITERATE

11. Automate Alerting and Incident Response
12. Use Data to Drive Decisions
13. Iterate and Improve Continually

BEFORE WRITING CODE

1. Identify and Empower a Product Owner

There must be an NGA product owner who is accountable as their primary role for how well the product meets user needs. The product owner must work with users and engineering and design leadership to balance tasks, set priorities, and make business, product, and technical decisions. The product owner is also responsible for ensuring that features are built and delivered according to well defined acceptance criteria and definitions of done. The product owner prioritizes features and backlogs and approves completed work.

The product owner's main job is ensuring the product's success, both initially and over time. A product owner is an individual (not an office or organization), can come from many places in NGA or the NSG, and must be familiar with major aspects of the product and what problems it is solving.

Example Questions to Answer

- **Who is the accountable product owner?**
- **How does the product owner interact and work with the program manager?**
- **Do all stakeholders agree that the product owner has the authority to make decisions about priorities, features and delivering functionality to users?**

2. Deeply Understand User Needs

Before beginning technical work, a product owner and team must engage with real and/or expected users to identify and understand their underlying needs, as well as learn how the product or service will fit into their work. This is not just hearing a user say what they "want.'' It requires deeply understanding their mission and goals as well as current approaches. Only by understanding the existing state and the reasoning behind it can the team design a solution. Teams need to identify key performance indicators (KPIs) to measure product success and value provided to users.

Human-centered design involves engaging with users to understand their problems and concerns. Methods for doing so include but are not limited to interviews and usability testing. These user needs, including KPIs and key findings, must be shared with stakeholders regularly and kept in a single source of truth for reference (generally Jira and Confluence). Product owners must set performance baselines for the old product or service if there was one, and document plans to improve user satisfaction.

Example Questions to Answer

- **Who are your users or user personas?**
- **What are the top user needs you are addressing?**
- **What is your roadmap for addressing these needs?**
- **What changes to user needs have you identified as a result of researching or testing (such as paper prototypes) with users?**
- **How have you mapped user journeys through the product and tracked them to identify areas of poor performance?**
- **What are your KPIs and how do they map to higher level NGA goals?**
- **What are you not doing? Why?**

3. Prioritize Rapidly Delivering Value to Users

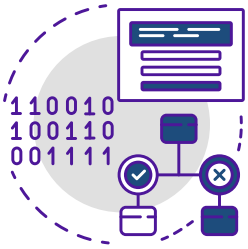
Once user needs are deeply understood, the team must decide which of those needs to prioritize. To do this, **the team must define a minimum viable product (MVP), writing it down and sharing this with key stakeholders. The MVP should be the smallest release possible to deliver a capability that solves a user need, or some part of one.** The goal is to get feedback, test assumptions, learn quickly, and tackle risk upfront. The product owner must break down the MVP into actionable work items that are easily understood by the entire team. Teams must gather feedback as early as possible using mockups or interactive prototypes. As feedback is gathered, the prototype's fidelity should mature and the team must transition to testing with the real product.

Every day that a product isn't actively solving user needs in production increases risk the team is building the wrong thing. Teams must prioritize rapidly getting an MVP to actual users in production to learn what works and how to successfully iterate; this first capability must be delivered in months, not years. **If the initially-defined MVP cannot be delivered to real users in production within single-digit months, it must be broken down further.** Which Enterprise services are able to be used is a key component

of accelerating delivery; teams should understand the options available (DevSecOps tools, Platforms-as-a-service, API management, etc.) and use these shared services to the maximum extent possible. See the Appendix for more details on these services. Additionally, if there is a commercial product that can meet this need, or a product or service exists at NGA that can be minimally altered to meet the user need, those should be prioritized over creating a new product.

Example Questions to Answer

- **What is included in the MVP? Can you describe it in an "elevator pitch"?**
- **Which user need will the MVP address?**
- **How will you test if the MVP will actually address that user need? How long do you expect it will be before the first capability is being used by users in production?**
- **How will you capture feedback from the MVP?**
- **How will you decide what to prioritize after the MVP is delivered?**
- **How do you weigh feedback from stakeholders and users with different priorities?**



WHEN BEGINNING AND THROUGHOUT DEVELOPMENT

4. Use Version Control

All source code required to deploy and run a given application, including both infrastructure-as-code and the application code itself, must be maintained in a government-managed version control system. This enables NGA and our partners to collaborate and provides NGA ownership and traceability of all code changes. It is expected that all source code will be peer reviewed and will pass the automated tests (which are also stored in version control) before being merged into the "main" branch.

An enterprise, government-managed version control instance is accessible on all security domains and must be used as the version control for all software teams. The current specific product and where to go to learn more can be found in the Appendix.

Example Questions to Answer

- **Is your code in a government-managed version control? Can we see it?**
- **What domain(s) is your code on and how does it move to higher domains?**
- **Is your code documented so that an engineer not on your team could successfully build and deploy it in the intended environment?**
- **What is your standard code review and merge request process?**
- **How can another team reuse parts or all of your code?**
- **What is your team's branching strategy? Are you using "protected branches" to control how changes get applied to critical branches?**

5. Track Work Consistently

Each team must have a central location for tracking its current and future work. All data in the work tracking tool must be accessible as needed to any NGA employee and owned by NGA, not by contractor teams. Any change to a project's source code must be tied to an issue that provides context for the change and enables the team to discuss and refine a requirement.

The government-managed Jira must be used as the work tracking tool and Confluence must be used to document design details and decisions. Work items in Jira should include both the item that needs to be worked on as well as acceptance criteria and/or a definition of done.

Example Questions to Answer

- **Do you track issues in Jira?**
- **How do you use Confluence?**
- **Are source code changes connected to issues in the work-tracking system?**
- **How do you categorize issues in the issue tracker (bugs, new feature, etc.) and use these categories when prioritizing work?**

6. Automate Testing

Changes to source code that impact system functionality must include automated tests that verify the functionality. **Unit, integration, and end-to-end tests must run automatically when code changes are proposed to the "main" branch in version control. Code linters and security vulnerability scans also must run on merge requests to the "main" branch.** If any of these tests or scans fail, merging the changes into the main branch must not be allowed. There must be automated accessibility tests for any user-facing functionality. Automated security reports from the continuous integration / continuous delivery (CI/CD) pipeline must be available to NGA security teams.

NGA's enterprise, government-managed CI/CD pipeline must be used for running tests and builds as well as scans for code quality and vulnerabilities.

Example Questions to Answer

- **Can you demonstrate running the automated test suite as part of your continuous integration process?**
- **What is your testing code coverage, and how has that changed over the life of the product?**
- **What code linters are you using?**
- **What product are you using for security vulnerability scans, how often are these run, and what do you do with the results? Are there gates in the pipelines to enforce specific thresholds?**
- **What accessibility testing do you have or do?**

BY THE FIRST DELIVERABLE THAT PROVIDES VALUE TO THE USER

7. Provide Support and Feedback Mechanisms

When users experience problems, they need clear and well-defined support paths for reporting them. T**eams must be able to measure support issues** to know if the service is working appropriately and to decide if their roadmap needs to consider bugs first or features of higher priority. User support is not only for identifying problems, however; **users must have a way to provide feedback to the team, and get responses to that feedback in a reasonable timeframe.**

Users must able to easily provide feedback or contact someone for help with an issue. For problems with a product, users should be able to contact a dedicated help desk or the Enterprise Service Center for support. For more general questions or feedback, these same mechanisms may be used or the team may have an email listserve or RocketChat room the team monitors to answer questions in real time. Support issues should be tracked and follow a tiered model, making it back to the development team for further assessment and prioritization by the product owner in the backlog. Bug or issue status in the team's work tracker (Jira) must be available to other NGA teams, users, and customers for transparency.

Example Questions to Answer

- **How do users contact someone for help if they are struggling with the product or service?**
- **What is your target response time for user support issues, and how have you met that time over the life of the product?**
- **How do user support issues make it back to the team? Provide an example of this working well.**

8. Document and Share APIs

APIs must be documented in industry standard API-Specification formats (i.e. OpenAPI, RAML), rather than PDFs or Word documents, and they must be accessible via self-service in a development environment with a standardized path to production. API documentation must be readily available and understandable to all developers at NGA. APIs should follow GEOINT standards and be able to be automatically tested for standards compliance.

APIs must be documented and registered on NGA's API Developer portal on all domains where the APIs are available. API design must follow the API Development guidance available at **https://devcorps.pages.gs.mil/Documentation/process/api_guidance.html**

Example Questions to Answer

- **Where is your API documentation? Please provide a link.**
- **Can other developers easily access and test out your APIs in a non-production environment on their own?**
- **How do people begin using your APIs in production?**
- **How many API consumers do you have and what are the primary use cases for these consumers?**
- **Is there a standard that your data aligns to, or is supposed to align to?**
- **How will new versions of your API roll out and old versions be deprecated?**
- **Who is the steward for the data behind the API and how do I talk to them if needed?**

9. Automate Deployments

New versions of a product must be automatically deployable to non-production and production environments. All infrastructure setup must be stored in source code version control (GitLab) as infrastructure-as-code. New servers must have programmatic stand up and no human should need to touch production servers to update a product. Automated production deployment of a new version must be scripted to generally happen without any downtime, and teams should be able to quickly roll back any change. This eliminates the need for scheduled maintenance and reduces the coordination needed between systems for production releases. Additionally, production systems need to be regularly and automatically backed up, and restoring from backup must be regularly exercised to ensure procedures are understood and working.

NGA's government-managed CI/CD pipeline must be used for automating deployments.

Example Questions to Answer

- **Can you demonstrate running the deployment as part of your CI/CD pipeline?**
- **What is your deployment frequency?**
- **Is your configuration management and infrastructure as code in version control?**
- **If a deployment causes an increase in errors, how do you learn about this and how do you roll back?**
- **Do deployments require downtime? If so, why? How are you decreasing this over time?**

10. Automate Monitoring

Automating monitoring allows you to collect performance data to have constant knowledge about how your product is performing and changing. Each member of the development and operations team must have visibility into how systems are working through application performance monitoring (APM). APM data collected must include metrics such as CPU and memory use, the latency of functions, and the number of errors. White-box (internal) and black-box (external) monitoring must be automated and continually running.

Products must use CIO-T's Secure Operations Group (TO) suite of enterprise monitoring tools that make metrics visible to development and operations teams as well as the Enterprise Service Center. Additionally, **performance data and metrics must be easy to discover and accessible by any NGA employee.**

Example Questions to Answer

- **What is your product's availability?**
- **How do you know if your product is up or down?**
- **What are the median and 90th percentile response times for the main actions users take on your product?**
- **How large is/are your database(s) and how fast are they growing?**
- **How did you decide the data and metrics you need to capture and where do you currently capture them from (or plan to capture them from)?**

AS YOU ITERATE

11. Automate Alerting and Incident Response

When an issue arises with a product, automatic alerts must notify an on-call engineer as well as the Enterprise Service Center. Teams need "runbook" documents that describe how common incidents should be handled, including escalation paths to communicate with product owners, dependent systems, and customers. Teams must have a place to communicate via chat in real time and pull in engineers from other systems, as needed. Teams must ensure all team members have proper access and equipment to triage incidents. Additionally, products are expected to be self-healing to the maximum extent possible. Automatic remediation steps, such as the infrastructure spinning up new application servers when CPU load is high, should be in place and regularly tested.

Products must integrate with CIO-T's Secure Operations Group (TO) enterprise monitoring tools that send alerts directly or through a Service+ workflow. Runbooks must have a consistent format, be updated regularly, and be maintained anywhere the incident response team has access to shared documents, such as Confluence. RocketChat is commonly used for collaboration across teams and during incidents.

Example Questions to Answer

- **What happens when your product or service goes down? How do you find out and who gets alerted (both people to fix the service and customers who are impacted)?**
- **Where do your outage runbooks live and when was the last time you used and/or updated them?**
- **What are the first actions outlined in your runbooks to identify and alleviate incidents?**
- **What are your incident response communications channels, both internally and externally?**
- **Does your team periodically (and before the initial launch) engage the Secure Operations Group and other stakeholders to review the incident response process and your runbook documents?**

12. Use Data to Drive Decisions

A team must have continuous ways to measure how a product is performing for users. These include both human communication and interactions, such as reports to a help desk about issues and automated insight mechanisms that track product use, such as events fired, logs, and usage funnels. Teams must use these metrics for decision making on the product backlog and roadmap, which must be accessible to users and stakeholders. Synthetic monitoring, which is available in most application performance monitoring tools, also should be used.

NGA's enterprise web-analytics software must be installed on browser-based applications and websites, unless there are alternatives built into the product that provide sufficient visibility of the data. Three types of product metrics should be defined by the product owner and collected: performance metrics that tell you if the product is doing what it is supposed to; business metrics that tell you how well your product supports organizational goals; and exploratory metrics that tell you how people are using your product and help you understand where to go next.

Example Questions to Answer

- **How are you collecting metrics? Which tools are in place to measure user behavior?**
- **What are your primary performance, business, and exploratory metrics? How are you measuring these and how have they performed over the life of the service?**
- **How do you use these metrics to make product decisions? Provide an example.**
- **How are your metrics shared?**
- **How do you collect qualitative data from users and stakeholders, and how does that data inform you decision making?**

13. Iterate and Improve Continually

Products and services are never "finished." Using agile methods means getting real people using your service as early as possible and making improvements throughout the lifetime of the product. Making improvements means more than doing basic maintenance like fixing bugs in code, deploying security patches, and keeping runbooks up to date; continuous improvement means you can capture and respond to changes in user needs, technology, or government policy throughout the lifetime of the product.

Iteration isn't just for the early stages of a product's development; a team must be able to make substantial improvements throughout the lifetime of the service. As product complexity grows, teams must focus on balancing paying down technical debt with new feature development, ensuring the product continues to be adaptable over time to respond rapidly to user needs.

Questions to Answer

- **How has your product evolved in the past six months?**
- **What are the biggest upcoming changes, and why are they needed?**
- **What is your product's design and engineering lead time?**
- **What is your product's development and deployment lead time?**
- **Can you share your product roadmap?**
- **What is your usability testing process and cadence? What changes have you made to the product based on usability testing?**

CONCLUSION

The above 13 elements, collectively referred to as "The NGA Software Way," ensure each NGA software product has the necessary people, processes, and tools in place to successfully deliver. Together they will enable NGA to consistently build, release, and operate software that meets our users' needs and expectations, and to deliver on our mission. As always, we will adapt and iterate on this document as we learn more.

APPENDIX

Current NGA Tools

NGA manages a suite of tools, referenced above, across all security domains for building, deploying, and operating software:

- NGA's enterprise, government-managed version control instance is GitLab.
- NGA's enterprise, government-managed CI/CD pipeline is available using both Jenkins and GitLab CI. For new projects, GitLab CI is recommended:
  - Within NGA's enterprise, government-managed CI/CD pipeline, there are a number of other tools. Fortify, OWASP ZAProxy, and SonarQube are used for security and quality checks, and Threadfix is used for aggregating and managing these scans.
  - Selenium and Puppeteer are commonly used tools within this pipeline for testing browser features and are recommended.
  - Terraform and/or CloudFormation running through this pipeline must be used to build infrastructure, and Ansible is frequently used for application deployments and configuration management.
- NGA's enterprise web-analytics software is Matomo.

As these specific tools change over time, such as if NGA chooses to use another tool with a similar or better capability, this Appendix will be updated. The current deployed version of each tool, as well as more details on these and other available tools, can be found on the DevX Portal: **https://devx.pages.gs.mil**