THE NGA SOFTWARE WAY

EXECUTIVE SUMMARY

SOFTWARE METRICS

To deliver on its current and future mission, NGA needs to deliver useful software faster and more consistently.

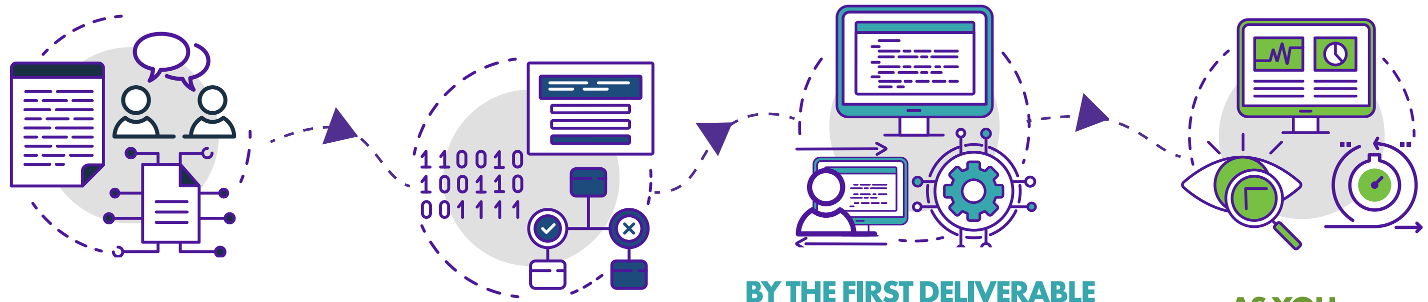
To measure progress toward this intent, NGA will track three key metrics for each software product: **availability, lead time for changes,** and **deployment frequency**. Additionally, each software product is expected to have its own **product-specific** metrics to track how well the product is working for its users. Together, these metrics balance speed and reliability and describe a product's impact and value.

AVAILABILITY | **LEAD TIME FOR CHANGES** | **DEPLOYMENT FREQUENCY** + **PRODUCT-SPECIFIC METRICS**

To consistently track these metrics and realize the above intent, this document provides a set of elements NGA and contractor teams must meet while building, iterating on, and operating software products. These elements ensure each software team has the people, processes, and tools in place to effectively operate at NGA.

The metrics and elements in this document, collectively referred to as "The NGA Software Way," serve as an implementation guide to supplement NGA's Technology Strategy and provide additional specific guidance for teams building and operating software.

THE NGA SOFTWARE WAY ELEMENTS



BEFORE WRITING CODE

1. Identify and Empower a Product Manager
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
7. Document and Share APIs

BY THE FIRST DELIVERABLE THAT PROVIDES VALUE TO A USER

8. Provide Support and Feedback Mechanisms
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

A NOTE FROM THE CTO

[illegible]

Alex Loehr
NGA Chief Technology Officer

To deliver on its current and future mission, NGA needs to deliver useful software faster and more consistently. 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 users and customers. The NGA Software Way is an implementation guide to supplement NGA's Technology Strategy and provide additional specific guidance for teams building and operating software.

Our intent is to get to a place where NGA software products are designed and built with users, can be automatically updated in production daily, and use product-specific metrics to make decisions. We believe in small, empowered, cross-functional teams that deeply understand their users and constantly work directly with those users to deliver on their mission.

Implementing The NGA Software Way won't happen all at once; we must start with iteratively tracking and improving these metrics and meeting each element below. As we do this together, and with an understanding of this intent, we will successfully deliver software products with the speed, accuracy, and precision NGA's mission demands.

SOFTWARE METRICS

The three initial metrics all programs must track for each software product delivered to users are availability, lead time for changes, and deployment frequency. 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 as well as 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 in this DORA report: https://services.google.com/fh/files/misc/state-of-devops-2019.pdf.

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

For software teams using NGA's enterprise-managed version control and Continuous Integration/Continuous Delivery (CI/CD) pipeline, deployment frequency and lead time for changes will be automatically tracked within those services' dashboards. To enable that automation, the recommended tagging method can be found in the appendix. For teams not using NGA's enterprise-managed version control and CI/CD pipeline, lead time for changes and deployment frequency must be independently calculated and transparently shared. All teams are expected to regularly share their metrics with teammates, stakeholders, users, and leadership. Eventually, teams will programmatically share these metrics to a central, NGA-managed location.

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, that product is unavailable. Availability may be impacted by actions a product team itself takes or through issues with upstream or downstream dependencies. There is no distinction in availability to users based on what caused the issue; 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 proactively. Although teams may have different availability requirements depending on mission and user needs, teams are generally expected to maintain at least 99.9% availability in production, which translates to about one business day of downtime per year.

LEAD TIME FOR CHANGES

Lead time for changes tracks how long it takes to deliver a feature for mission use once it has been developed. Lead time for changes is calculated as the median time between when code is committed to when it is deployed to a production environment, where it can be used during an actual mission (not in a test environment). That figure is measured across all commits within a given time period. Measuring this enables teams to understand where bottlenecks are in getting features to production and prioritize where to optimize.

Shorter lead times are better because they enable faster feedback on what is built and allow 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. DORA research shows high-performing teams have lead times for changes of less than one week.

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, however, where users actually use the product to complete the mission or function. To the maximum extent possible, deployments are expected to be automated (see Automated Deployments on page 9), so increasing deployment frequency does not linearly increase workload on the team.

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). DORA research shows high-performing teams deploy at least once per week.

PRODUCT-SPECIFIC METRICS

In addition to the above three metrics that will be consistently tracked across all software programs, each software product is expected to have its own key performance indicators (KPIs) and metrics to track how well the product is working for its users. While the first three metrics are about balancing speed and stability, these product-specific metrics are about product priorities and value. Software products contribute to NGA's mission in different ways, so their success must be uniquely measured, and each team must define its own product-specific metrics.

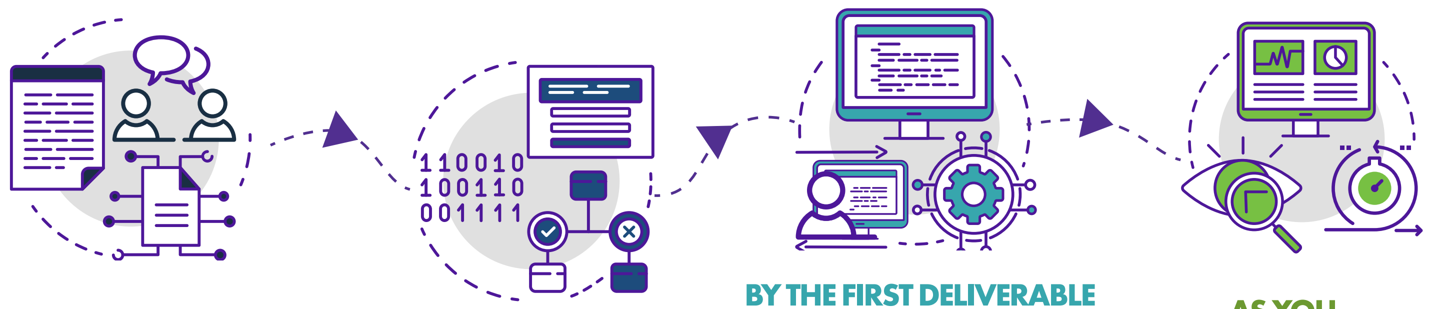
Because these product-specific metrics will not be consistent across NGA, they cannot be defined at the organizational level. However, each product will include: (1) performance metrics that describe if the product is doing what it's supposed to; (2) business metrics that convey how well the product is meeting organization and mission goals; and (3) usage metrics to understand how people are using the product. Product-specific metrics give software teams a measurable way to evaluate decisions and inform their priorities. Stakeholders and leadership can use this data to logically determine support and resource needs and assess the product's progress and impact. Product managers must define and regularly share these product-specific metrics with teammates, stakeholders, users, and leadership.

These metrics can be collectively referred to as the 3+1 metrics — availability, lead time for changes, deployment frequency + product-specific metrics. Together, they provide an understanding of the software product's speed, stability, impact, and value.

THE NGA SOFTWARE WAY ELEMENTS

To consistently track these metrics and realize the above intent, the rest of this document provides a set of elements NGA and contractor teams must meet while delivering software products. These metrics and elements, collectively referred to as "The NGA Software Way," ensure each software product has the people, processes, and tools in place to 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.** The NGA Software Way applies to software products of all phases, including existing software products across the agency (not simply new products). Except where noted below, these elements must also be used when evaluating and implementing commercial software products for use at NGA. We understand that for large, existing software products, implementing The NGA Software Way won't happen all at once; it should start with tracking the metrics above and iteratively meeting each of the elements below.

"Product" is an overloaded term at NGA. In this document, a "product" is software at any stage of development, rather than a finished intelligence product. Products can be directly user-facing or indirectly user-facing like backend services that users depend on but only developers directly interact with. 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 across all domains.



BEFORE WRITING CODE

1. Identify and Empower a Product Manager
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
7. Document and Share APIs

BY THE FIRST DELIVERABLE THAT PROVIDES VALUE TO A USER

8. Provide Support and Feedback Mechanisms
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 Manager

Each software product must have a dedicated product manager who is accountable for delivering a product that meets user needs. The product manager works with users and the engineering, design, and program teams. The product manager is also responsible for ensuring features are built and delivered according to well-defined acceptance criteria and definitions of done. The product manager prioritizes features and backlogs and approves completed work.

The product manager's main job is ensuring the product's success at each stage of development, from the initial idea phase through product launch. A product manager 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 which problems it is solving. **The product manager is accountable for creating the product vision, metrics, and roadmap, as well as defining user personas.** The product manager must also conduct regular research to understand the product's landscape, evaluate how other services are solving similar problems, and consider whether adopting an existing solution is a better path than building a new one.

Government product managers are critical to the success of delivering the right capabilities at the speed of relevance. NGA is investing in ways to accelerate the growth of product management skills and establishing a set of standards for collateral like roadmaps, metrics, visions, and user personas. Product managers work as a counterpart to program managers and directly with developers, whether government employees or contractors, to ensure customer needs are met.

Example Questions to Answer

- **Who is the accountable product manager?**
- **How does the product manager interact and work with the program manager?**
- **Do all stakeholders agree the product manager has decision-making authority on which features get delivery priority?**
- **How does the product manager work with developers?**

2. Deeply Understand User Needs

Before beginning technical work, a product manager and team must engage with real and/or expected users to identify and understand users' underlying needs, as well as learn how the product or service will fit into the users' work. This is not just a government product manager (or anyone else) hearing users 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 and engineer a solution. Teams need to identify KPIs to measure product progress, success, and value provided to users. Before a product is live, the product manager also must work with the team to establish metrics that will help evaluate progress. Initially, there can be a small number of metrics with the expectation that after the product launches and matures, the number of metrics grows along with team members' understanding of how their solution addresses user needs.

Human-centered design involves engaging with users to understand their problems and concerns. Methods for doing so include interviews, usability testing, and prototyping. The identified user needs, including KPIs and key findings, must be regularly shared with stakeholders and kept in NGA's enterprise-managed collaboration tools. Product managers must assess performance baselines for the old product or service, if applicable, and document plans to improve user satisfaction in collaboration with program managers and the engineering and design teams.

Example Questions to Answer

- **What is the product vision?**
- **Who are your users or user personas? How have you engaged with them, and what user research have you done?**
- **What are your product-specific metrics? How do they map to higher-level NGA goals?**
- **What changes to product priorities have you identified as a result of researching or testing (such as paper prototypes) with users?**

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), either in written form or, ideally, as a set of wireframes, and then share it with key stakeholders, customers, and users. The MVP is the smallest actual deliverable that can be reviewed by real customers to get feedback on if it solves a need and to test assumptions, learn quickly, and decrease risk upfront. The product manager 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.

Each day a product isn't actively solving user needs in production increases the risk the team is building the wrong thing. Teams must prioritize rapidly getting a product 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. Using NGA's enterprise services is a key component of accelerating delivery; teams should understand the options available (DevSecOps tools, Platforms-as-a-service, Application Programming Interface (API) management, etc.) and use these shared services to the maximum extent possible. See the appendix for more details on these services. **Market research must be conducted to understand if a solution already exists inside or outside of NGA that sufficiently meets user needs or can be easily altered to do so. Only after thorough analysis of the existing landscape can software product teams confidently determine whether to buy, augment, or build a solution to address user needs.**

Example Questions to Answer

- **What is included in the MVP? Can you describe it in an "elevator pitch?"**
- **Which user needs does the MVP address? How will you test if the MVP will actually address those user needs?**
- **How long do you expect it will take for the first capability to be used by users in production?**
- **How do you weigh feedback from stakeholders and users with different priorities?**
- **What was your market research strategy, and can a commercial product meet this need instead of building custom software?**
- **What are you not doing? Why?**

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 an enterprise-managed version control system. This enables NGA and our partner teams to collaborate and provides NGA ownership and traceability of all code changes. It is expected that all source code is commented on in a consistent manner and is peer reviewed and passes the automated tests (which are also stored in version control) before being merged into the main branch. The current suggested versioning method and how to learn more can be found in the appendix.

An enterprise-managed version control instance is accessible on all security domains and must be used as the version control for all teams building software at NGA. We understand this does not apply to commercial software NGA does not directly develop, although infrastructure-as-code for managing or deploying this commercial software should still live in the enterprise-managed version control to facilitate automated testing and deployments. The current specific product and where to go to learn more can be found in the appendix. Mirroring repositories to higher domains is also supported in this enterprise-managed version control to enable teams to more easily build-low, move-high.

Example Questions to Answer

- **Is your code in the enterprise-managed version control? Can we see it?**
- **What domain(s) is your code on, and how does it move to higher domains?**

(continued on next page)

- **Is your code documented so an engineer not on your team could successfully build and deploy it in the intended environment using only information in version control?**
- **What is your standard code review and merge request process?**
- **What is your team's branching strategy? Are you using "protected branches" to control how changes get applied to critical branches?**
- **What is your team's versioning method?**

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 enterprise-managed work tracking and collaboration tools must be used to document features to be built, defects to be fixed, design details, and decisions by teams building software at NGA. We understand this does not apply to commercial software NGA does not directly develop, although enterprise-managed collaboration tools should still be used for knowledge sharing and documenting decisions. Work items in the enterprise-managed work tracking tool should include both the item that needs to be worked on and acceptance criteria and/or a definition of done. When applicable, NGA enterprise-managed templates should be used for documenting items like MVPs and solution epics.

Example Questions to Answer

- **Where do you document team decisions?**
- **How do you track work tasks and progress?**
- **Are source code changes connected to issues in the work-tracking tool?**
- **How do you categorize issues in the work-tracking tool (defects, new features, 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. Each commit should trigger a builder of the software as well as automated tests that provide feedback in minutes. **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. Software product teams should use pre-approved container configurations wherever possible. Tests must be continuously created, maintained, and executed as code is developed, and there must be automated accessibility tests for any user-facing functionality. Automated security reports from the CI/CD pipeline must be available automatically and on demand to NGA security teams.

NGA's enterprise-managed CI/CD pipeline must be used for running tests and builds as well as scans for code quality and vulnerabilities by teams building software at NGA. We understand some testing may not apply to commercial software NGA does not directly develop, although security scans and other capabilities within the enterprise-managed CI/CD pipeline may still be valuable to expedite approvals and decrease lead time for changes. Within the automated testing process, all software, including open-source software, must be scanned and cross-referenced with known vulnerabilities. NGA is investing in ways to accelerate this process by minimizing manual intervention, and the appendix to this document will be updated as changes to those processes occur.

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?**
- **What automated API testing do you have or do?**

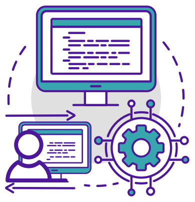
7. Document and Share APIs

APIs must be documented in industry standard API specification formats (i.e., OpenAPI), rather than Adobe PDFs or Microsoft 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 must follow relevant GEOINT standards and allow for automatic testing 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/standards/api_guidance.html (Note: you must have a CAC to access this page).

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? Where is this documented?**
- **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 does an API consumer talk to that person if needed?**



BY THE FIRST DELIVERABLE THAT PROVIDES VALUE TO A USER

8. Provide Support and Feedback Mechanisms

When users experience problems, they need clear and well-defined support paths for reporting them. Teams must be able to measure support issues to know if the service is working appropriately and decide if their roadmap needs to prioritize defects or features first. 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 be 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 in-product solution, email listserv, or chat 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 manager in the backlog. Defect or issue status in the team's work tracker 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.**
- **Is your issue escalation process documented, and does it include when or how to reach the product's domain experts?**

9. Automate Deployments

New versions of a product must be automatically deployable to non-production and production environments. Automation is crucial to reducing deployment risk in all environments; the automated deployment process in test environments should not differ from the one used in production deployments. All infrastructure setup must be stored in NGA's enterprise-managed version control 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 must be regularly and automatically backed up, and restoring from backup must be regularly exercised to ensure procedures are understood and working.

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

Example Questions to Answer

- **Can you demonstrate running a 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?**
- **When a product becomes not deployable (failing builds or tests, for example), how does fixing that get prioritized against new feature development?**
- **Do deployments require downtime? If so, why? How are you decreasing this over time?**
- **What are the largest impediments to fully automating deployments, and how are you working through them?**

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 a product is 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 telemetry — metrics, logs, traces) and black-box (external behavior) monitoring must be automated and continuously running. It is the external monitoring that must be used to calculate the availability metric, as it doesn't matter to users if the product is "working" if they are unable to access it.

Products must use CIO-T's Secure Operations Group (TO) suite of enterprise monitoring tools, which 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 are your databases, and how fast are they growing?**
- **How did you decide the data and metrics you need to capture, and from where do you currently capture them (or plan to capture them)?**

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 managers, 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 TO's enterprise monitoring tools, which 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 NGA's enterprise-managed collaboration tools.

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)?**
- **What are the first actions to be taken when you identify various incidents? Where is this documented?**
- **What are your incident response communications channels, both internally and externally?**
- **How does your team engage with TO 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, should also 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 must be defined by the product manager and collected: performance metrics indicate the product is performing correctly; business metrics indicate how well your product supports organizational goals; and usage metrics show how the product is used and inform future development needs.

Example Questions to Answer

- **How are you collecting metrics? Which tools are in place to measure user behavior and customer satisfaction?**
- **What are your primary performance, business, and usage 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 with stakeholders, teammates, and leadership?**

13. Iterate and Improve Continually

Products and services are never “finished.” Getting real people using your product as early as possible, monitoring the product’s performance, and making improvements throughout the lifetime of the product enable teams to adapt quickly. Making improvements means more than doing basic maintenance like fixing defects in code, deploying security patches, and keeping runbooks up to date; continual 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 a product’s lifetime. 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?**
- **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 metrics and 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 enable NGA to consistently build, release, and operate software that meets our users’ needs and expectations and deliver on our mission.

NGA’s CIO-T organization and leadership are accountable for The NGA Software Way’s adoption and success. Members of NGA’s CTO Office act as dedicated members of NGA leadership to work across teams and track The NGA Software Way implementation through mechanisms like program management reviews, product reviews, surveys, and monitoring tools.

As always, we will adapt and iterate. This document will be updated and republished as both NGA and industry best practices evolve.

APPENDIX

Current NGA Tools

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

- ► NGA's enterprise-managed version control instance is GitLab.
  - NGA recommends using the Semantic Versioning method to version application source code. Specific guidance is available at https://devcorps.pages.gs.mil/Documentation/process/versioning_code.html#semantic-versioning (Note, you must have a CAC to access this page).
- ► NGA's enterprise-managed CI/CD pipeline is available using both Jenkins and GitLab CI. For new projects, GitLab CI is recommended.
  - GitLab requires environment tags like `environment: production` so it knows to automatically track production activity like deployment frequency and lead time for changes. Specific guidance is available at https://docs.gitlab.com/ee/ci/environments/#deployment-tier-of-environments.
  - Within NGA's enterprise-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.
  - Terraform and/or CloudFormation are used though this pipeline to build infrastructure, and Ansible is frequently used for application deployments and configuration management.
  - NGA recommends using:
    - – ESLint in conjunction with the AirBnB JavaScript Style Guide (https://github.com/airbnb/javascript) to standardize JavaScript development.
    - – Pylint in conjunction with the PEP 8 Style Guide (https://www.python.org/dev/peps/pep-0008) to standardize Python development
- ► NGA's enterprise-managed work tracking and collaboration tools are Jira and Confluence. Additionally, RocketChat is commonly used for collaboration across teams and during incidents, although it tends to be best for ephemeral conversations rather than as a long-term source of truth for documentation.
- ► NGA's enterprise web-analytics software is Matomo: https://demand-analytics.dev.gs.mil/home/ (Note, you must have a CAC to access this page.)

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/ (Note, you must have a CAC to access this page.)

NGA.mil