Digital Government Authority
هيئة الحكومة الرقمية

Guideline

Development and Operation (DevOps)

26, February 2023

Document Type: Guideline

Document Classification: Public

Issue No.: 1.0

Document No.: DGA-1-2-2-208

Table of Content

1 Introduction

Digital Government Authority works to enhance digital performance within government entities, raise the quality of services provided and improve the beneficiary experience, in line with the strategic directions of the Digital Government Authority. The authority has prepared a "guideline for the development and operation approach" to increase the efficiency of digital services and products, by adopting a cooperative or joint approach to accomplish the tasks performed by the application development and IT operating teams in the entity. The DevOps approach is the most advanced way to develop and maintain IT systems, combining end-to-end ownership with engineering and automation. This guide is one of the references that will contribute to promoting the adoption and use of the DevOps approach in support of digital transformation activities in the government sector. In addition, this guide is part of a series of guidance documents aimed at supporting and promoting digital excellence.

2 Guideline Objectives

This guideline aims to:

- Encouraging government agencies to implement the DevOps methodology, which in turn will
  - o Enhancing communication and interaction between software developers and operators and project management experts
  - o Reduce costs and improve productivity at a faster rate
  - o Execute related processes more reliably and efficiently
- Understanding DevOps methodology requirements for effective implementation.
- Learn to effectively measure the adoption of DevOps methodology

3 Guideline Scope

The scope of the guide to the methodology for developing and operating DevOps software includes seven main sections, which include an overview of the methodology, the purpose of its application, its use cases, and the most prominent stages of its development, in addition to the most important requirements for its application and adoption, and the end of the mechanism for measuring its sustainability.

4 Target audience

This guideline targets experts, practitioners and technical staff working in software/system development in government entities..

5 Guideline Statement

5.1 Development and Operations (DevOps)

In the past, most software development has proceeded in a “waterfall approach” with a strict separation of the phases of the software development as well as the operation of software (e.g., administration, patching, and maintenance). This traditional divide has caused several key challenges:

- Missing ability to react to beneficiaries feedback.
- Missing collaboration between the different teams as part of the software development to ensure overall success.
- Overly complex and interfaces between the different steps of the software development.
- Missing focus on the stability and the run of software versus the development of new features and assets.

These different challenges have led software developers to reconsider the traditional ways of developing software and to adopt new ways of working It included achieving the following:

- **Single team mindset**: moving away from silo-based processes and interactions towards end-to-end delivery and collaboration in fully empowered products teams.
- **Equality of non-functional requirements**: involvement of the right experts and measurement of the reliability of software to ensure proper and resilient operation of the software.
- **Continuous improvement**: moving away from manual rework and finger pointing towards a continuous improvement of the existing software.

- **Fact-based decision**: moving away from decision being taken by misaligned KPIs and non-fact based decisions towards basing all decision on software development on data and proper aligned analysis.
- **Sharing culture**: moving away from individual developments towards cross-team collaboration and alignment to ensure mutual success.
- **Engineering practices**: Move away from customized and hard-to-reproduce assets and promote properly engineered solutions.
- **Automation**: moving away from manual controls and manual monitoring towards automation of the controls and monitoring to ensure that products become first-time-right.

5.1.1 DevOps Purpose/Value

The specific purpose of DevOps is It included achieving the following:

1. Generate faster results that are more tailored to beneficiaries.
2. provide a better experience and more clear direction to the development teams.

This is specifically required caused by Digital transformation. Digital is changing customer behaviors and technology expectations, while exponentially increasing pressures on IT demand, costs, and delivery. This is also generating the value of DevOps as the enablement of Digital transformations is a key requirement to stay competitive and meet citizen demands in the future.

The Omni-channel experience is now the norm, creating new expectations for digital delivery. Citizens are now hyper-informed, with new products, services, and features made continuously available to them through mobile devices and apps. Furthermore, citizens expect service and interaction 24/7, placing increasing pressure on IT services.

These expectations are increasing the pace and complexity of software and infrastructure change. Server and storage demand, for example, have risen 10 times because of the vast volumes of new data and real-time machine learning analytics. The cost to deliver and maintain digital solutions has increased by two to three times, due to duplicative systems, costly legacy architecture, and dual maintenance support. To adapt to rapidly changing customer expectations while maintaining quality, government entities are implementing DevOps.

The value lies in the application of the DevOps methodology compared to the traditional methods related to the following:

- **Faster time to market**: from typically 15-50 days for deployments to production to 15-20 minutes (typically due to automation of deployment and the software build and test)
- Higher efficiency: a productivity increase of development teams of typically 30% (typically due to more efforts being first-time-right)
- **More stability**: Typically, a reduction of critical incidents by around 70% (typically due to better monitoring and higher code quality)
- Greater employee satisfaction, as a culture of autonomy and self-steering.

5.1.2 DevOps Cases

The government sector is slowly adopting the principles of the "DevOps" approach, and in general, most government entities have recently begun to explore the "DevOps" approach, and are still using the (old) waterfall methodologies in developing most software products. Examples of adopting the "DevOps" approach in government entities include:

- Large-scale projects: Several of the large-scale software projects in the government sector are currently using DevOps already as a method for improved efficient delivery project delivery. Specifically for software developments where the final goal is ambitious and less clear (e.g., development of a national identity solution, Citizen Apps, national platforms, etc.).

- Resilient operation: The important of adopting DevOps arise where resiliency in operation is a key. It is crucial to ensure changes do not impact the operations of the actual product.

- Frontend development: Provision of updated software front-ends (eg for mobile applications for government services) ranks as a model candidate for experimental use of DevOps principles due to frequent changes and quick visibility for faster implementation.

5.1.3 DevOps Phases

The ISO/IEEE have defined clear phases for the rollout of DevOps within an organization as part of ISO 32675. It defines the different phases of the DevOps introduction as well as the execution of the phases. In the following, this section summarizes the insights both from a perspective of ISO standard as well as best practices that have been observed in the industry.

DevOps requires a significant mindset shift and upskilling of the workforce instead of just implement new processes and tools. This mindset shift requires significant attention and change management.

The different phases of the DevOps implementation process include:

1. Plan: create a plan for how to develop the application/ the service and its further development.
2. Create: develop the actual code.
3. Verify: manage and ensure the quality of the delivered code and further end products.
4. Package/ pre-prod: make the product/ service ready to be deployed.
5. Release: ensure that code can be delivered to production.
6. Configure: configure the application and the underlying infrastructure components.
7. Monitor: monitor the underlying health of application and the hardware usage.

1. Plan

Planning includes the definition and planning referring also to the business value as well as the possible application of the software.

Key activities:

- Collection of production metrics (including SLAs/service-level objectives [SLOs]) and feedback.
- Requirements definition (use case, prototyping).
- Collection and definition of business metrics.
- Update of release metrics.
- Definition of release plan (timing, business case).
- Definition of security policy, adherence, and requirements.
- Identification and role definition for key stakeholders from business, IT, software development, and architect teams.

Example of "Plan" Tools:

- Atlassian Jira
- Atlassian Conference
- Azure DevOps
- Trello
- Pivotal Tracker
- Basecamp
- Monday.com

2. Create

The creation phase includes the actual coding, building and the actual configuration of the software product so that all the software development process should be completed.

Key activities:

- Design of the software and architecture (software, configuration)
- Code, code merge, code quality, and performance
- Build and monitor build performance
- Perform functional tests
- Release software to production

Example of “Create” Tools:

- GitHub
- GitLab
- CircleCi
- Atlassian Bamboo
- AWS CodeBuild
- Jenkins

3. Verify

Ensure the quality of software releases, code quality, and deployment of high quality software for production.

Key activities:

- Performance of acceptance tests
- Performance of regression tests
- Performance of static analysis (quality and compliance)
- Performance of security analysis (vulnerability)
- Conducting performance tests
- Conducting configuration tests

Example of “Verify” Tools:

Test automation:

- Selenium
- Jest
- Lamdatest
- Jasmine
- TestComplete
- Pytest

Static analysis:

- SonarQube
- Raxis

- SmartBear Collaborator

Test lab/ security:

- OWASP
- Fuzz
- Wapiti
- w3af
- SonarQube

4. Package/Pre-prod

Involves the activities once the release is ready for deployment, also referred to as packaging or staging.

Key activities:

- Management to get the required Approvals/pre-approvals
- Release package configuration
- Management of automated releases (for example, application release is automatically pushed)
- Management of triggered releases (for example, upon a "ready state" being met)
- Release staging and holding

Example of "Package/Pre-prod" Tools:

- Azure DevOps
- Atlassian Bamboo
- Jenkins
- CircleCi
- Codeship

5. Release

Includes scheduling, orchestration, provisioning, and deploying software into production and targeted environment. This aspect of the toolchain includes application release/deployment automation and release management.

Key activities:

- Release coordination
- Management of scheduled/timed releases
- Deploying and promoting application
- Management of change controls
- Management of fallbacks and recovery

Example of "Release" Tools:

- Azure DevOps
- GitLab
- Ansible
- Chef
- AWS CodeBuild

6. Configure

Includes activities for infrastructure provisioning, configuration of additional IT, post software deployment.

Key activities:

- Infrastructure provisioning and configuration (including storage, database, and network)
- Application provisioning and configuration
- Solutions that facilitate above activities are tools around continuous configuration automation, configuration management and infrastructure as code

Example of "Configure" Tools:

- Chef
- Helm
- Ansible
- Terraform

- Puppet Labs
- Docker

7. Monitor

Includes application performance monitoring, infrastructure monitoring, and alerting tools.

Key activities:

- Performance and availability of the IT infrastructure, network, and application
- End-user response and experience
- Production load metrics gathering
- Often provides inputs to ‘plan’ activities required for changes, new release cycles

Example of “Monitor” Tools:

- New Relic
- Splunk
- Amazon CloudWatch
- Sumo Logic
- Google Cloud
- Prometheus
- Sensu

5.1.4 DevOps Requirements

To run DevOps at scale and capture its full potential, there are several prerequisites that need to be in place. It is typically implemented in two phases :top-down and bottom-up phase. The objective of the top-down phase is to define the overarching vision and create the “case for change” in the entity, while the bottom-up phase concerns the actual implementation of DevOps methods/tools in the individual product or development teams.

Checklist for the top-down phase:

- An organizational model needs to be in place that shows the development teams as well as their integration with the operations teams
- The organizational and operating model needs to clarify both aspects: the reporting lines as well as the performance appraisal process including all key team members from business, development as well as operations
- Appropriate architecture principles (e.g., interfaces via APIs) need to be formulated and a plan must be developed to realize the target system architecture for the systems needed to implement DevOps (key aspects of the target architecture are further elaborated below)
- A plan needs to be in place for the adoption of DevOps tools and a rollout plan must be developed including required customizations
- The required talent from an infrastructure, development as well as operations perspective for the bottom-up pilots needs to be available and a plan for further hiring needs to be established.

The system architecture for DevOps can be characterized by different enablers that come together. The enablers are:

1. **Collaboration platforms include** the source code management, work item management, the documentation as well as the management of incidents. It also includes tools to chat and perform post-mortem management as well as the required dashboards and reporting.
2. **The delivery pipeline** and the operations pipeline include the different tools directly supporting the DevOps phases that we have elaborated on above.
3. **Management tools** include asset and financial management tools.
4. **PaaS services** include the direct provisioning of infrastructure as databases, middle ware and focused specific tools for APIs.

5. **Infrastructure provisioning** includes the relevant tools for the provisioning of storage, network and servers.

Further developments during the bottom-up phase need to be completed.

Checklist for the bottom-up phase

- A DevOps playbook needs to be defined including the key processes (development, finance, capacity and portfolio planning, infrastructure management etc.) and their respective steps in the future DevOps organization.
- An engineering support team set up to drive and support the DevOps transformation.
- The required capacity for further pilots needs to have been established and a further hiring plan for capacity is set up.

After successful completion of these two phases, the government entity can move into the actual adoption of DevOps within the government entity. This entails two key cultural shifts that are required to truly see the required changes in the team-operating model.

Culture of ownership

Teams run products as small businesses with the following qualities:

- A single team mindset: Autonomous teams are accountable for a clearly defined product and cover the full IT value chain, including deployment and operations.
- Treating development, quality, security, scalability, and reliability as equally important.
- Continuously improving its members, product, and ways of working.
- Taking fact-based decisions and practicing a “measure everything” philosophy.
- Best practices and learnings are shared internally and externally in a blameless way, open-source technologies are leveraged, and contributions are shared back.

Technical practices and automation

Teams use modern engineering practices and push automation by doing the following:

- Using modern engineering practices to make collaboration within and between teams easier.
- Taking an “automate everything” approach.
- Adopting fully automated testing and a continuous delivery system

These techniques typically lead to an increased productivity.

5.1.5 DevOps Adoption

After completing the top-down and bottom-up phases, the government entity can move into the onboarding of teams to the DevOps methodology. To build DevOps capabilities quickly and at scale, government entities should define a repeatable approach.

A successful DevOps roll-out combines both ‘top-down’ and ‘bottom-up’ approach. It over-invests in cultural transformation, as this is the most challenging part of the DevOps journey. While the weighting of top-down vs. bottom-up initiatives is highly dependent on the context, they should include the following best practices:

Top-down approach

The top-down approach follows a series of steps:

- Determine target design, based on key design choices.
- Define vision and aspiration in line with business objectives.
- Establish structured and frequent communication using a variety of formats (for example, townhall, team visits), coaching and role modelling of management based on “servant leadership” principles.
- Implement capability building programs covering both formal training and informal events to upskill existing talent, recruitment of cross-functional and/or DevOps experts.

Bottom-up approach

In the bottom-up approach, DevOps is implemented team-by-team in a repeatable way, following a series of steps:

- Pilot the rollout approach team-by-team based on business priorities and potential impact.
- Follow structured roll-out and/or scale-up plan to cover full government entity, structured approach to develop tools/best practices in teams.
- Implement a pre-defined set of technical changes executed as a continuous improvement journey.
- Adopt approach for pilot applications: Most government entities focus on demonstrating impact with selected practices for three to five pilot applications first, then later extend the toolbox iteratively and prioritize in relation to the 'pain' of increasingly frequent releases.

Pilots

Pilots are essential to test the approach in the top-down approach before the bottom-up approach is executed at scale. Team-by-team transformation is based on business priorities and/or potential impact. The pilots adopt the following two steps as essential best practices:

- Shifting the culture by aligning objectives across development and operations teams, raising awareness about equal importance of the stages in the app's lifecycle, and sharing best practices (for example, in code development) and learnings.
- Aligning and integrating engineering practices and automation in the team by agreeing on approaches to code development, building the foundations to integrate automated testing, and starting to continuously deliver with one-click automated releases to beneficiaries.

Roll-out roadmap

It is essential for the rollout to build internal coaching capabilities on both "soft" and "hard" automation skills by creating sufficient internal knowledge to support the adoption of DevOps. This needs to be owned also by internal resources.

The rollout typically happens in 4 phases. The top-down planning typically not exceed 4-6 weeks whereas the pilot and bottom-up planning can also partially overlap but can extend over 4-6 months up to multiple years depending on the scope. The rollout can afterwards be extended over time. It also requires taking into account the required technology prerequisites to roll out the model to the entire government entity.

It should be noted that the model for the rollout of DevOps should be unified within in each government entity. Keeping multiple models makes alignment more difficult and creates unnecessary interfaces. The adoption fostered through two practices that are observed as essential for successful rollout :

1. **A high intensity learning process**

This process aims to increase DevOps capabilities within teams. This could include dedicated time for improvements in 2 to 4r acceleration sprints.

Typically the preparation include acceleration sprints with the team to ensure:

- ✓ Clarity on aspiration for the overall software development in terms of both functional and non-functional requirements
- ✓ Availability of the right resources (min. 50 percent of team time)
- ✓ Availability of required DevOps and Cloud engineers

The acceleration sprints then according to best practice proceed via first the introduction to the working practices. These working practices include the process design, the change management as well as key processes as incident management as well as the development itself.

Similarly, it is common to see that the acceleration sprints should cover the relevant key elements of tooling and technology introducing the relevant tools that will be introduced as well as the key aspects thinking about the development processes. These include specifically the coverage of automated testing as well as working with a CI/CD pipeline. It should also include a clear introduction to the architecture target picture as well as the roadmap to get there. Finally, during the acceleration sprints the team should already work together and show key parts of the agile ceremonies as e.g., the sprint demos for the content that they have learned.

2. Team capability model

The government entity would also need to establish the capability to work as a team between the development and the operations teams. Building this capability requires proceeding in phases and requires handing over responsibilities to the DevOps teams in phases. In terms of capability building, a common model is that the capability built by changing the operating model in phases.

The first phase is the co-location of the development and operations teams. In this phase agile working practices are in place for development and the relevant automations for testing and code quality checks are implemented. However, the tracking of dependencies and the deployment still require manual controls and approvals. This means that development and unit as well as acceptance testing become part of the DevOps team whereas deployment and release as well as all additional infrastructure and operations tasks will remain separate.

In the second phase the ops engineers become part of the team. This allows to integrate the operations tasks like incident management and user support into the team. Similarly, the testing and monitoring now be fully automated so that the ops engineer can take the responsibility for the deployments alone. This means that also in terms of responsibility the DevOps Teams become responsible for development, unit testing as well as acceptance testing and also deployment and release. Monitoring, incident management, as well as business continuity remain part of the infrastructure and operations teams.

Finally, the third phase is fully agile where a full integration within the team and the processes is achieved. Relevant controls and monitoring are fully automated end-to-end so that also the teams can take full ownership of the deployments. This means that also in terms of responsibility the DevOps Teams become responsible for development, unit testing as well as acceptance testing, deployment and release, application operations and monitoring, incident management as well as business continuity management. Only database management, platform management as well as infrastructure operations and monitoring remain with the respective infrastructure and operations teams.

The consecutive execution of these phases also leads to a full end-to-end toolchain that automates the relevant controls. This includes specifically that in a full DevOps model all code should be handled in one
repository. Checking in code into the repository also trigger automatically that relevant tests and security checks are executed.

Similarly, when executables need to be built the build process needs to have been automated with relevant tools and further integration, user and security tests need to be fully automated. Finally, also the actual deployment of the executable should be an automatic process in the relevant toolchain. The complete toolchain described here is therefore also referred to as the DevOps toolchain.

The adoption of the model in a best-practice setup cross-checked as part of the rollout on a continuous basis. Typically, there are cross-checks against the principles of the aligned approach every 6 months .

5.1.6 DevOps sustainability & measurement

There are six top-level KPIs, which are typically used in a DevOps context to measure the overall performance of IT.

KPI	Description	Top-class performance	Average class performance	Underperforming / legacy
Lead time	Time it takes from "code committed" to "running in production"	<1 hour	1 week – 1 month	>1 month
Deployment frequency	Frequency of which new increments rolled out to production	Multiple per day	1/week – 1/month	<1/month
Mean time	In the event of a	<1 hour	<1 day	>=1 day

taken to recover	malfunction, time it takes to restore normal operation			
Change Failure Rate	Percentage of erroneous deployments to production	0-15%	15-30%	>31%
Productive time	Share of developers over total staff	>80%	50-80%	<50%
Employee Net Promoter Score (NPS)	Likelihood of employees to recommend their place of work	>70	40-70	<40

Table 1 : KPIs to measure performance of IT

These different KPIs as well as the respective benchmarks values listed above to ensure the overall performance of a government entity IT. The entire value stream monitored to facilitate continuous improvement. This part both reviews within the teams as well as the review process within the DevOps

organization every 6 months. It is essential to set up a corresponding governance around the reviews and the corresponding definition of actions.

A standard practice is that the quality department within the organization can take over the responsibility of conducting the reviews against the ISO standards and define corrective actions if required.

The operational performance inputs can be gathered from the tools directly whereas for the further inputs interviews with the teams are required. Concerning the measurement of input metrics, the measurement of the required metrics needs to be set up from the tools and monitoring tools need to be configured. Also, input data as the number of stories and other need to be gathered from the tooling for sprint planning etc.

This allows monitoring each part of the DevOps process previously described including the number of stories produced by each team, their respective complexity, the time it takes to build, the errors in production etc. This data can be further leveraged to drive the continuous improvement process. This excludes the planning process as this one defines several of the baselines for the metrics described below.

Metrics to measure in a DevOps setup:

1. **Create**:
   - Build-success rate (%): Share of successful (software) builds of code committed in the cost repository
   - Velocity (SP's/day): Volume of story points that a team or an individual developer can deliver per day
   - Code quality: Quality of the software code as measured by software code quality tools

2. **Verify**:
   - Unit test coverage (%): Percentage of code committed that is covered by unit tests
   - Build success rate (%): Share of successful (software) builds of code committed in the cost repository
   - Security test pass rate (%): Share of committed software code that passes security checks as executed by specialized code security tools
   - Test success rate (%): Share of tests that are successful for committed code

3. **Package/ pre-prod:**
   - Package success rate (%): Share of packages successfully produced

4. **Release**
   - Deploy success rate (%): Share of deployments that are successful for committed code

5. **Configure**
   - Mean time to restore: Average time it takes for an application to restore normal operations after a (significant) incident

6. **Monitor**:
   - Incidents per month (per priority): Number of production incidents by priority/severity

The measurement of these metrics and the definition of required corrective measures belongs to the teams. Additionally, formal reviews following the ISO standard conducted, as per best practice, every 6 months and presented to the CIO of the government entity to define and execute corrective measures.

6 Table of Definitions

The following terms and expressions - wherever they appear in this document - shall have the meanings indicated on the opposite side of each of them, unless the context requires otherwise

Term	Description
Authority	Digital Government Authority
Government Entities	Ministries, authorities, public institutions, councils, national centers including any additional form of a public entity.
Digital Government	Promotes administrative, organizational and operational processes between the various government entities in their transitioning to a comprehensive digital transformation to allow easy and effective access to government digital information and services.
Beneficiary	Citizens, residents, visitors, government agencies, private sector, non-for-profit sector, inside or outside the KSA that require to interact with a government entity to receive any of the services offered in the Kingdom.
DevOps	DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This

	speed enables organizations to better serve their customers and compete more effectively in the market.
Digital Transformation	Digitally and strategically transforming and developing business standards and models that would rely on data, technologies, and ICT.
Stakeholder	Parties and entities that affect and are affected by decisions, directions, procedures, objectives, policies and initiatives of the digital government and share some of their interests and outputs and are affected by any change that occurs in them
CI/CD	CI/CD is a method to frequently deliver applications to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment.
Waterfall software development	The waterfall methodology is a project management approach that emphasizes a linear progression from beginning to end of a project. This methodology, often used by engineers, is front-loaded to rely on careful planning, detailed documentation, and consecutive execution.
Service Level Agreement (SLA)	A service-level agreement (SLA) sets the expectations between the service provider and the customer and describes the products or services to be delivered, the single point of contact for end-user problems, and the metrics by which the effectiveness of the process is monitored and approved.
Plan	Create a plan for how to develop the application/ the service and its further development
Create	Develop the actual code
Verify	Manage and ensure the quality of the delivered code and further end products
Package/ pre-prod	Make the product/ service ready to be deployed
Release	Ensure that code can be delivered to production
Configure	Configure the application and the underlying infrastructure components
Monitor	Monitor the underlying health of application and the hardware usage
Reporting line	Disciplinary reporting line in an organization
Performance appraisal process	Employee performance review process as part of HR performance management
Collaboration platforms	Tools within the DevOps architecture to facilitate the collaboration between teams
PaaS services	Services for providing certain platforms as e.g., pre-configured databases
Infrastructure provisioning	Ensuring actual deployment of infrastructure as servers and network for required software services

Portfolio planning	Process to plan IT investments and/or IT projects in an organization.
servant leadership	Servant leadership is a leadership approach built on the belief that the most effective leaders strive to serve others, rather than accrue power or take control. The aforementioned others can include customers, partners, fellow employees, and the community at large.
Agile ceremonies	Agile ceremonies are meetings where a development team comes together to keep each other updated on their project's details. At the same time, other Scrum ceremonies, such as the sprint retrospective, helps the scrum team look back on their work and find ways of improving for future sprints.
Functional Requirements	Those requirements would include an analysis of the core requirements described with specific use cases and key metrics that define their success and validate if the technology considered either meets or has limited or no use for the functional requirements defined. Additionally, critical functional requirements that need to be considered are the suitability of the ET's architecture and alignment with the governmental entity's current and envisioned architecture in the future.

7 Table of Abbreviations

Abbreviation	Description
CI	ntegrationI Continous
CD	eliveryD ontinousC
E2E	End-to-end
SRE	Site Reliability Engineering
DevOps	Development and operations combined
DGA	Digital Government Authority
ISO	International Standards Organisation
IEEE	Institute of Electrical and Electronics Engineers
NPA	Net promoter score
SP	Story Point
CIO	Chief Information Officer

هيئة الحكومة الرقمية
Digital Government Authority

DGA.GOV.SA